UAH

Implementación hardware de un algoritmo de correlación eficiente de códigos CSS multinivel

Máster Universitario en Sistemas Electrónicos Avanzados. Sistemas Inteligentes

Departamento de Electrónica

Presentado por:

D. Santiago Murano

Dirigido por:

Dra. Mª Carmen Pérez Rubio

Dr. Enrique García Núñez

Alcalá de Henares, Julio de 2014

Resumen

En el presente trabajo de fin de máster se describe la implementación en hardware de un correlador eficiente de Conjuntos de Secuencias Complementarias (CSS, *Complementary Sets of Sequences*) Multinivel, empleados en sistemas de sensado activo basados en CDMA (*Code Division Multiple Access*). Gracias a las propiedades ideales de las sumas de correlaciones aperiódicas, las secuencias CSS son cada vez más utilizadas en los sistemas basados en CDMA.

Una de las ventajas de los CSS es el empleo de generadores y correladores eficientes los cuales requieren menos operaciones comparados con una arquitectura directa.

Empleando las arquitecturas existentes para correladores de CSS binarios, se ha generalizado para su empleo con un alfabeto multinivel logrando así secuencias con valores reales. Esto permite obtener las siguientes ventajas: el aumento del número de longitudes que pueden generarse y correlarse, y la mejora eliminando las limitaciones de las arquitecturas previas en el número de secuencias en el conjunto. Una de las aplicaciones de estas secuencias multinivel, es la generación se secuencias ternarias (3 niveles), con bajos valores de Relación de Potencia Pico a Potencia Media, PAPR (*Peak-to-Average Power Ratio*), con el fin de mejorar la eficiencia energética de los amplificadores de potencia en las etapas de emisión.

Al momento de diseñar la implementación hardware, se estudió el funcionamiento de los generadores y correladores eficientes, para luego diseñar un modelo de simulación, en el cual se analizaron varias formas de gestión de la cuantificación y truncamiento de los datos para implementar la solución que lleva a una menor degradación de la correlación. El diseño para la implementación hardware se realizó en VHDL para ser implementado en un dispositivo FPGA (*Field-Programmable Gate Array*).

A Iva y mi familia.

Agradecimientos

Quiero agradecer en primer lugar a mis directores, Mª Carmen y Enrique, por todo el esfuerzo realizado en estos últimos meses para poder acabar antes de mi regreso, los constantes empujones y ánimos que me dieron para que no baje los brazos. Sin ese constante apoyo hubiese sido difícil poder finalizar.

A Carlos De Marziani por alentarme e involucrarme en esto, y haberme incentivado a vivir una experiencia espectacular del otro lado del charco. He aprendido muchas cosas, tanto académicas como personales, lo que me ha hecho crecer como persona.

Un extensivo agradecimiento a los integrantes del grupo de Investigación GEINTRA de la UAH. Por la cálida integración brindada en el grupo, con las populares reuniones, ricas en conocimiento y en degustaciones locales. Así como también a mis compañeros del fondo 22, Joaquín y César, por compartir buenos ratos en los cafes de media mañana, el aporte de sus consejos y experiencias, e introducido en el mundo del LaTeX.

A los MUSEOS, siempre dándome ánimos y ofreciendo su ayuda y calidez haciéndome sentir como en casa. Por los maravillosos encuentros que hemos tenido dentro y fuera del ámbito del máster. Ha sido un placer conocerlos, un grupo maravilloso con el que siempre despejarse después de momentos duros de estudio.

Un amplio agradecimiento a mis compañeros de la residencia, que prácticamente han pasado a formar parte de mi familia. Gracias por todo el apoyo y fuerza brindados, la compañía en las noches de estudio, y las salidas turísticas compartidas.

Un enorme agradecimiento a mis amigos tanto de Trelew como de Comodoro, siempre dándome ánimos, preguntando por mi trabajo y mi estancia en el extranjero. Gracias por aguantar mi poca comunicación y mis despistes.

Sin duda quiero agradecer eternamente a mi familia por el apoyo y el aliento constante. A mis viejos, por el apoyo incondicional, por bancarme siempre en todas las decisiones que he tomado, a pesar de que en los últimos meses haya estado poco comunicado, sin ellos todo esto no hubiera sido posible.

Por último, agradecer al amor de mi vida Ivalú, quien me acompaño a lo largo de esta maravillosa experiencia, aguantando mis ánimos y mis llegadas tarde. Siempre pendiente de mi, liberándome de tareas para dedicarme exclusivamente al máster. Y por su ayuda en correcciones finales, este trabajo es tanto de ella como mío. Gracias por ser la maravillosa persona que sos, con tu humor y cariño eternos.

Muchas gracias a todos, sin “vosotros” esto hubiese sido imposible.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El empleo de esquemas de codificación con buenas propiedades en su función de autocorrelación son de un gran interés en aplicaciones basadas en técnicas CDMA (*Code Division Multiple Access*), en el caso de sistemas RADAR (*Radio Detection and Ranging*) [HG88, LCZ10], aplicaciones de ensayos no destructivos (NDT) [WC92], sistemas de comunicación OFDM [DJ99] o técnicas de segunda generación de difusión terrestre [HWY$^+$11].

La eficiencia de las aplicaciones basadas en técnicas CDMA dependen de las propiedades de los códigos que se utilicen [FD96]. Esto es: propiedades ideales de auto-correlación (AC) con picos elevados para un desplazamiento nulo del código y lóbulos laterales de la función de AC cercanos a cero. Así mismo, es necesario que los valores de correlación cruzada (CC) entre los diferentes códigos sean lo más próximos a cero. El cumplimiento de estas propiedades ayudan a la minimización de las interferencias inter-símbolo, ISI (*Inter Symbol Interference*) y por acceso múltiple, MAI (*Multiple Access Interference*). Los códigos CDMA tradicionales, como los Gold [Gol67] o las secuencias Kasami [Kas66], presentan valores de lóbulos laterales no nulos en la AC y CC, lo que implica una disminución del rendimiento en muchas aplicaciones reales, donde son muy comunes las emisiones aperiódicas, la detección asíncrona, el efecto multi-camino, etc.

Dado que con una sola secuencia es imposible eliminar por completo y de forma simultánea los lóbulos laterales de las funciones de AC y CC [Wel74], muchos autores se basan en conjuntos de secuencias complementarias, CSS (*Complementary Sets of Sequences*). Los códigos complementarios fueron considerados por primera vez por M. Golay en [Gol61], y consisten en un par de secuencias binarias con la misma longitud, cuya suma de AC y CC cumplen con las propiedades ideales. Sin embargo, sólo provee dos pares mutuamente incorrelados lo que no lo hace viable para sistemas con más de dos emisores.

Unos años más tarde, Tseng y Liu [TL72] propusieron los conjuntos de K secuencias complementarias (K-CSS), siendo éstos una generalización de los pares Golay, donde el número K de secuencias que componen el conjunto puede ser mayor que dos. Con lo que eliminando la restricción en el número de secuencias a generar, produce un aumento en la ganancia de proceso, y también más de dos conjuntos mutuamente incorrelados. Además estos códigos pueden generarse y detectarse mediante correlación usando algoritmos eficientes que reducen el número de operaciones frente a una implementación directa. Específicamente, en [MUH$^+$07], De Marziani *et al.* amplían el trabajo de F. J. Álvarez *et al.* en [ÁUM$^+$04] y proponen una arquitectura modular para la generación y correlación eficiente de conjuntos complementarios de secuencias (ECSS), el cual fue más tarde implementado en [PUH$^+$06] por M. C. Pérez *et al.* No obstante, el número de secuencias a generar de estas arquitecturas K-CSS está limitado a $K = 2^k$, y la longitud de los códigos a $L = K^Q$ $(k, Q \in \mathbb{N} - \{0\})$, siendo Q el número de etapas modulares que tiene el generador/correlador.

Darnell y Kemp extendieron el concepto de CSS al alfabeto multinivel en [DK88, KD82], donde proponen un algoritmo no recursivo para generar CSS multinivel (valores reales) sin las limitaciones de longitud de secuencias (L), ni del número de secuencias por conjunto (K). Posteriormente, en [GUG13] se propone una arquitectura de generadores y correladores eficientes para conjuntos de secuencias complementarias multinivel utilizando algoritmos recursivos. Con estos algoritmos se puede generar K conjuntos CSS multinivel, con $K \in \mathbb{N} - \{0, 1\}$.

Actualmente, las secuencias CSS se utilizan también como base para la creación de secuencias ortogonales generalizados, GO (*Generalized Orthogonal*), usadas/empleadas en sistemas CDMA cuasi-síncronos (QS-CDMA). Ejemplos de códigos GO son las secuencias LS (*Loosely Synchronous codes*) [SBH01], secuencias aperiódicas ZCZ (*Zero Correlation Zone*) [PUH+09] o secuencias GPC [GUG+12].

Debido a la necesidad de aplicaciones que se ejecuten en tiempo real, se requieren correladores eficientes que satisfagan los requisitos del sistema, es por ello y las múltiples aplicaciones de los conjuntos complementarios multinivel, que se ha decidido implementar el correlador eficiente estudiado en [GUG13]. Para ello se analizaron las modificaciones necesarias al correlador ECSS binario para poder llevarlo al alfabeto multinivel, y de ese análisis se realizó un modelo de simulación para poder determinar la mejor manera de realizar dichas modificaciones de una forma eficiente.

Posteriormente, en virtud de los resultados obtenidos en simulación, se analizaron 5 opciones de truncamiento, así como también el comportamiento del correlador para diferentes valores de retardo y amplitud. En este trabajo se diseñó el correlador en VHDL, para ser implementado en un dispositivo FPGA (*Field-Programmable Gate Array*), por lo que es importante conocer y fijar ciertos límites a la hora de definir el número de bits por etapa, así como también la cantidad de multiplicadores a utilizar para aprovechar al máximo los recursos de la plataforma.

1.1. Objetivos

Se propuso en este trabajo el estudio de la cuantificación y el truncamiento del bus de datos limitados por el posible hardware a utilizar, y la implementación en hardware configurable de arquitecturas eficientes para correlar códigos CSS multinivel, de modo que el procesamiento en tiempo real de longitudes largas sea posible. Luego, a partir de estos códigos CSS multinivel, pueden generarse códigos LS multinivel.

1.2. Contexto

Este trabajo se ha realizado en el Departamento de Electrónica de la Universidad de Alcalá, en el grupo de Investigación GEINTRA, que se encuentra especializado en el área de Sistemas de Localización y Posicionamiento. Específicamente, el trabajo se desarrolló en el marco del proyecto CODECSS (CCG2013/EXP-080).

1.3. Estructura

El presente trabajo se ha dividido en cuatro bloques:

- Estado del arte y aspectos generales.

  En esta sección se describirán aspectos generales de las técnicas CDMA y los esquemas de codificación, poniendo especial énfasis en los CSS. Posteriormente se describirán los

trabajos realizados en lo que respecta a algoritmos de correlación eficiente de CSS. Y por último, una breve descripción de algunas plataformas para implementación hardware.

- Conjunto de Secuencias Complementarias Multinivel

  Aquí se detallarán más específicamente los algoritmos y las arquitecturas existentes para la generación y correlación de Conjuntos de Secuencias Complementarios Multinivel, a partir de la generalización de algoritmos eficientes CSS binarios, y su generalización hacia CSS multinivel. También se incluye una de las principales aplicaciones de los CSS multinivel, como lo es la generación/correlación de pares complementarios ternarios óptimos.

- Implementación HW del correlador eficiente CSS multinivel

  En este apartado se estudiaran y describirán las consideraciones a tener en cuenta para el diseño del correlador. Se abordan aspectos relacionados con la cuantificación, y se plantean diferentes opciones de truncamiento de datos. A partir de los resultados obtenidos de dicho análisis, se plantea el diseño del correlador a implementar, se mencionan sus componentes y cómo funcionan, y por último se valida el diseño comparando simulaciones temporales de la implementación en VHDL, con un modelo de simulación del correlador realizado en MATLAB.

- Conclusiones y trabajos futuros

  Aquí se resume y comentan las conclusiones obtenidas a partir de los estudios realizados durante este trabajo. Seguidamente, se proponen futuros trabajos a realizar a partir de lo obtenido en este trabajo.

Capítulo 2

Estado del Arte

2.1. Aspectos generales sobre codificación

La aplicación de una función de correlación tiene por objetivo la determinación del grado de similitud entre dos señales. Así, la correlación es una medida de la dependencia de una señal con otra o consigo misma. Dada una familia A con K códigos binarios de longitud L $\{A = a_k[l] \in \{-1, 1\}; 0 \leq k \leq K-1; 0 \leq l \leq L-1\}$ la función de autocorrelación, ACF (*Auto-Correlation Function*), discreta periódica se define como (2.1) cuando $k = s$ y la función de correlación cruzada, CCF (*Cross Correlation Function*), periódica cuando $k \neq s$:

$$R_{a_k,a_s}[\tau] = \sum_{l=0}^{L-1} a_k[l] \cdot a_s[l+\tau] \tag{2.1}$$

En la expresión anterior las secuencias a_k y a_s son periódicas, es decir:

$$a_k = (\ldots, a_k[0], a_k[1], \ldots, a_k[L-1], a_k[0], a_k[1], \ldots, a_k[L-1], \ldots) \tag{2.2}$$

por lo que la suma $l + \tau$ se realiza en *módulo L*. La función de correlación aperiódica viene definida, sin embargo, por (2.3).

$$C_{a_k,a_s}[\tau] = \begin{cases} \sum_{l=0}^{L-1-\tau} a_k[l] \cdot a_s[l+\tau], & 0 \leq \tau \leq L-1 \\ \sum_{l=0}^{L-1-\tau} a_k[l-\tau] \cdot a_s[l], & 1-L \leq \tau < 0 \\ 0 & |\tau| \geq L \end{cases} \tag{2.3}$$

Cuando se cumple la condición de que las secuencias k y s son iguales ($k = s$) se obtiene la ACF aperiódica, y cuando son distintas ($k \neq s$) se obtiene la CCF aperiódica. Normalmente, se tiene sólo en cuenta la función de correlación aperiódica en el rango $0 \leq \tau \leq L-1$.

Un criterio ampliamente extendido para evaluar la calidad de una familia de códigos es el cálculo de su cota máxima de correlación (2.4) (*bound* en inglés):

$$\theta = max\{\theta_{AC}, \theta_{CC}\} \tag{2.4}$$

Donde θ_{AC} representa el valor del máximo pico lateral obtenido de las auto-correlaciones de los K códigos de la familia, y θ_{CC} es el pico máximo de las funciones de correlación cruzada.

$$\theta_{AC} = max\left\{\frac{|R_{a_k,a_k}[\tau]|}{R_{a_k,a_k}[0]}; \forall k \in [0, \cdots, K-1]; \forall \tau \neq 0\right\} \tag{2.5}$$

$$\theta_{CC} = max\left\{\frac{|R_{a_k,a_s}[\tau]|}{R_{a_k,a_k}[0]}; \forall k, s \in [0, \cdots, K-1]; k \neq s; \forall \tau\right\} \tag{2.6}$$

Si en las expresiones anteriores (2.5) y (2.6) se sustituye R_{a_k,a_s}, R_{a_k,a_k} por C_{a_k,a_s} y C_{a_k,a_k}, respectivamente, se obtienen las cotas para el caso de las funciones de correlación aperiódica.

Los códigos ideales, tienen un pico con una amplitud muy elevada en la ACF para $\tau = 0$ y lóbulos laterales nulos en $\tau \neq 0$ como se indica en (2.7). Además se trataría de códigos incorrelados entre sí, es decir, que su CCF sería cero para todos los posibles desplazamientos, lo que eliminaría cualquier interferencia entre ellos.

$$R_{a_k,a_s}[\tau] = \begin{cases} L, & \tau = 0, & k = s \\ 0, & 1 \leq \tau \leq L-1, & k = s \\ 0, & 0 \leq \tau \leq L-1, & k \neq s \end{cases} \tag{2.7}$$

Sin embargo, no existen familias de códigos unitarios que cumplan con éstas características simultáneamente. Es por eso que surgen los conjuntos complementarios, cuya suma de las correlaciones de un determinado número de secuencias logra este comportamiento ideal, no así las correlaciones individuales de las secuencias que componen el conjunto. En estos casos, cada emisor tiene asignadas más de una secuencia, utilizándose esquemas de modulación o reordenación de bits que permiten la transmisión de las mismas de una manera sencilla. No obstante, estos esquemas de modulación o reordenación de bits implican la aparición de interferencias en las ACF y CCF [MUH$^+$06]. Se conocen como interferencias inter-símbolo (ISI) a los lóbulos laterales no nulos que aparecen en la ACF, mientras que las interferencias producidas por la correlación cruzada no nula de dos códigos transmitidos por un mismo canal, se las conoce como interferencias por acceso múltiple (MAI). Para disminuir las interferencias ISI y MAI, se deben seleccionar aquellos códigos que tengan cotas de correlación, definidas en 2.4, lo más bajas posible.

Se dice que dos códigos son ortogonales cuando la ACF para $\tau = 0$ tiene un pico de amplitud L, y la CCF es nula para $\tau = 0$, es decir:

$$R_{a_k,a_s}[\tau] = \begin{cases} L, & \tau = 0, & k = s \\ 0, & \tau = 0, & k \neq s \end{cases} \tag{2.8}$$

$$\eta = \frac{1}{L}\sum_{l=0}^{L-1} |a_k[l]| \leq 1 \tag{2.9}$$

Cabe destacar, que la expresión (2.8) sólo es nula para el desplazamiento cero, lo cual es algo difícil de conseguir, ya que para ello el emisor y receptor deben estar perfectamente sincronizados. Además, existen efectos de reflexión que producirían que los códigos se recibieran en distintos instantes de tiempo, lo cual arruinaría la ortogonalidad entre códigos. Este caso, en que la señal puede propagarse por distintos caminos (debido a la reflexión de objetos o del suelo), se conoce como propagación multicamino.

Debido a que no existen secuencias con propiedades de correlación ideales simultáneamente en las funciones de auto-correlación y correlación cruzada, los esfuerzos se han centrado en encontrar secuencias ortogonales generalizadas, GO (*Generalized Orthogonal*) o cuasi-ortogonalidad generalizada (GQO), ya que estas secuencias pueden reducir las ISI y MAI en entornos cuasi-síncronos. Se dice que dos códigos son ortogonales generalizados si cumplen las características de la siguiente expresión:

$$R_{a_k,a_s}[\tau] = \begin{cases} \eta L, & \tau = 0, & k = s \\ 0, & 1 \leq |\tau| \leq W, & k = s \\ 0, & 1 \leq |\tau| \leq W, & k \neq s \end{cases} \tag{2.10}$$

Donde W representa una zona alrededor del origen donde las funciones de correlación tienen valor cero. A medida que sea mayor W, mejores características tendrán los códigos. Si $W = 0$, se obtienen los códigos ortogonales convencionales. Esta zona es conocida como zona de correlación zero, ZCZ (*Zero Correlation Zone*) o ventana libre de interferencias, IFW (*Interference Free Window*). Concretamente, la IFW se refiere al área de tamaño $2W + 1$ alrededor del origen. El parámetro η será uno cuando $a_k \in \{-1, 1\}$, aunque si $a_k \in \{-1, 0, 1\}$, es decir contiene valores nulos en el código, entonces η será menor que la unidad.

Para el caso de la cuasi-ortogonalidad generalizada, GQO (*Generalized Quasi-Orthogonal*), las interferencias alrededor del origen no son cero, sino que se encuentran debajo de un umbral ϵ aceptable según convenga en el sistema. De esta manera, se consigue aumentar el número de códigos que cumplen esta condición, aceptando un conjunto de interferencias controladas.

En la Figura 2.1, se pueden ver la ACF y la CCF aperiódica de una secuencia GO con una IFW bien diferenciada. Nótese que los lóbulos laterales de correlación fuera de la ventana IFW son grandes, por lo que es importante asegurarse que el retardo máximo entre usuarios sea menor que W y limitar la zona de interés a la IFW.

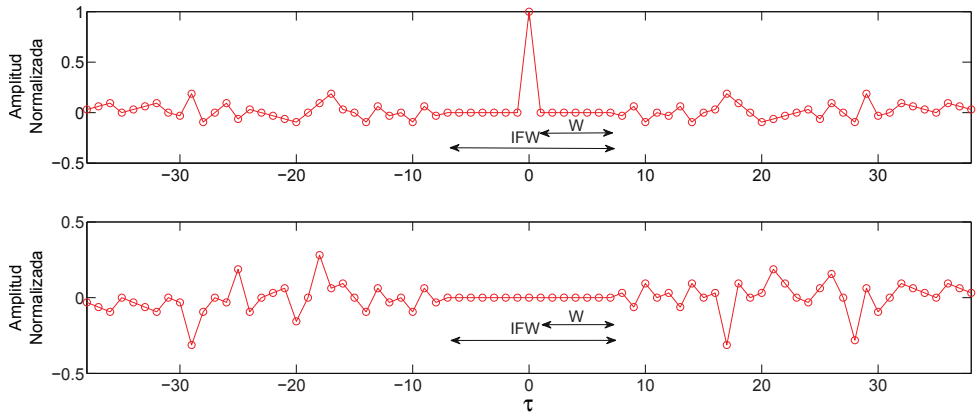


Figura 2.1: Función de correlación aperiódica de una secuencia tipo GO normalizada.

Como se observa en la Figura 2.2, las secuencias utilizadas en CDMA se pueden clasificar en dos grandes grupos, por un lado las secuencias reales, donde se encuentran las secuencias binarias como las Golay o las CSS. Por otro lado, están las secuencias polifásicas, que no se tratarán en este trabajo.

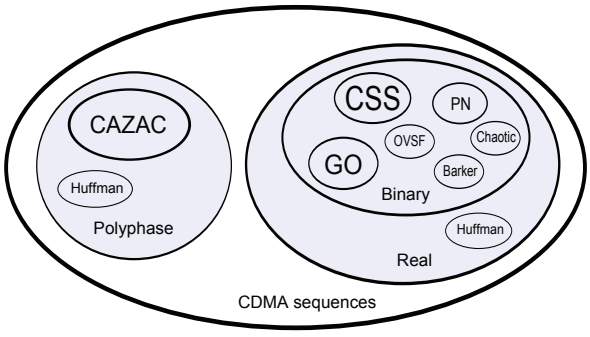


Figura 2.2: Clasificación general de secuencias utilizadas en CDMA [Gar13].

2.2. Conjuntos de secuencias complementarias CSS

En 1961 M. Golay analizó pares de secuencias cuya suma de funciones de autocorrelación aperiódica, SACF (*Sum of Aperiodic auto-Correlation Function*), es una delta de Kronecker, y la suma de las funciones de correlación cruzada, SCCF (*Sum of Aperiodic Crosscorrelation Functions*), es cero para cualquiera de los desplazamientos τ [Gol61]. Estas secuencias son conocidas actualmente como pares de secuencias Golay. Sin embargo, el número de conjuntos incorrelados sólo son dos, y las longitudes a generar están acotadas a $L_{|Gol} = 2^M \cdot 10^N \cdot 26^P$ siendo $M,\ R$ y P enteros positivos o cero.

En 1972, Tseng y Liu expandieron los pares Golay a los conjuntos CSS binarios de K secuencias complementarias, K-CSS (*Complementary Set of K Sequences*) [TL72], donde el número de secuencias K que componen el conjunto es mayor a dos (en el caso de las Golay) siendo $K = 2^k$, donde $k \in \mathbb{N} - \{0\}$. Como ventaja, los K-CSS disponen de hasta K conjuntos incorrelados entre sí.

Dado un conjunto $\mathbf{S}_j^{(k,q)}$ con K secuencias binarias $s_{j,i}^{(q)}$ de longitud L, se dice que:

$$\left\{\mathbf{S}_j^{(k,q)} = s_{j,i}^{(q)}[l] \in \{-1,1\};\ 0 \leq l \leq L-1;\ 0 \leq j,i \leq K-1\right\} \tag{2.11}$$

es un conjunto complementario si la suma de sus funciones de auto-correlación (SACF) es nula para todos los desplazamientos distintos de cero, y tiene un valor máximo de $K \cdot L$ para el caso de desplazamiento nulo.

$$SACF = \sum_{i=0}^{K-1} C_{s_{j,i}^{(q)}, s_{j,i}^{(q)}}[\tau] = K \cdot L \cdot \delta[\tau] = K^{Q+1} \cdot \delta[\tau] \tag{2.12}$$

En donde $\delta[\tau]$ es la función delta de Kronecker, y $L = K^Q$ la longitud de cada una de las $K = 2^k$ secuencias que componen el conjunto, siendo $k, q, Q \in \mathbb{N} - \{0\}$.

Dados dos CSS, $\mathbf{S}_{j,i}^{(k,q)}$ y $\mathbf{S}_{j',i}^{(k,q)}$, se dice que uno es compañero del otro si se cumplen las siguientes propiedades:

- Ambos conjuntos tienen K secuencias, y todas tienen la misma longitud L.
- La suma de las funciones de correlación cruzada (SCCF) de las secuencias de ambos conjuntos es nula para cualquier desplazamiento:

$$SCCF = \sum_{i=0}^{K-1} C_{s_{j,i}^{(q)}, s_{j',i}^{(q)}}[\tau] = 0, \qquad \forall\tau \tag{2.13}$$

Para evitar interferencias ISI y MAI, los lóbulos laterales de la ACF y los valores de la CCF deben ser nulos. Utilizando conjuntos K-CSS, se consiguen lóbulos laterales nulos sumando las ACF y CCF de las respectivas secuencias que componen dicho conjunto. En la Figura 2.3 se puede ver la suma de correlación para un conjunto de 4 secuencias.

Posteriormente, a partir de estas secuencias de conjuntos complementarios, se pueden obtener códigos binarios unitarios [Pér09], tales como los LS, ortogonales generalizados o macrosecuencias, obtenidos mediante la concatenación o entrelazado de secuencias K-CSS.

Uno de los inconvenientes de los conjuntos binarios CSS, son sus limitaciones en el número de secuencias y la longitud del código a generar. Limitación que con los conjuntos CSS multinivel no ocurre [GUG13]. Además, permiten la generación de secuencias CSS multinivel con bajo PAPR (*Peak-to-Average Power Ratio*) y longitudes flexibles [Pér09]. Más adelante se describirá el funcionamiento del generador y el correlador multinivel.

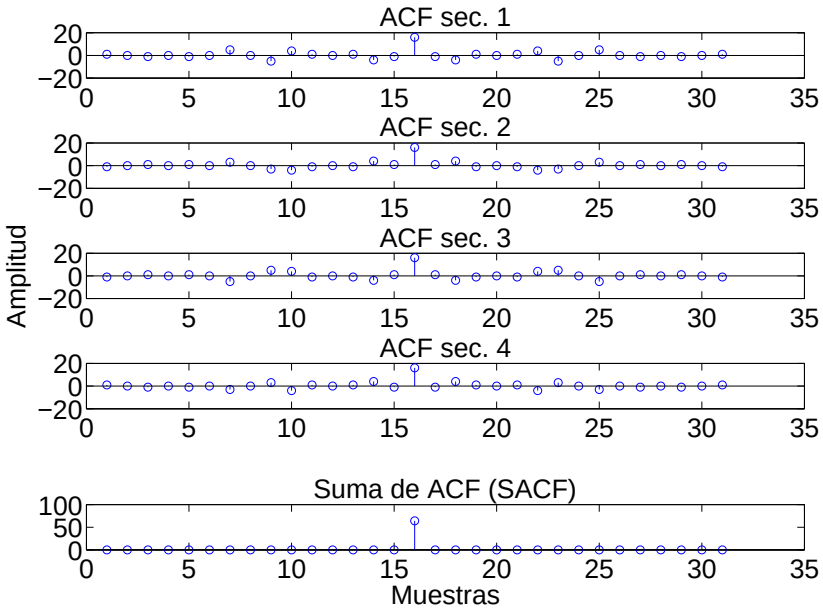


Figura 2.3: Suma con propiedades ideales de la correlación de 4 secuencias CSS de longitud $L = 16$.

2.3. Algoritmos eficientes de correlación CSS

La utilización de secuencias de gran longitud o del tipo CSS con varias secuencias por usuario, requieren arquitecturas eficientes, ya que la implementación de generadores o correladores directos conllevan una elevada carga computacional cuya implementación práctica puede llegar a superar los límites de velocidad, impidiendo el funcionamiento en tiempo real, y por tanto requerir hardware más costoso. Es por eso que los algoritmos deben diseñarse de manera que su complejidad sea mínima y los tiempos de ejecución se ajusten a las restricciones impuestas para el funcionamiento en tiempo real. Por tanto, son necesarios esquemas de correlación eficientes que disminuyan el número de operaciones por cada muestra de entrada.

Existen múltiples trabajos sobre este tipo de esquemas eficientes para secuencias pseudo-aleatorias, por ejemplo para secuencias-m [Bud89] o secuencias Gold ortogonales [Pop97]. En 1991 Budisin [Bud89] y posteriormente Popovic [Pop99] proponen un correlador Golay Eficiente (EGC, *Efficient Golay Correlator*). Este algoritmo permite simplificar el proceso de detección para el caso de utilizar parejas Golay de longitudes potencia de dos, $L_{|Gol} = 2^Q$ con una arquitectura eficiente. En la Figura 2.4 se puede ver el diagrama de bloques del correlador eficiente de pares Golay (EGC). Los valores $D^{(q)}$ representan retardos y los coeficientes $W^{(1,q)}$ la semilla de generación del par Golay, pudiendo tomar valores ± 1 [Bud89, Pop99].

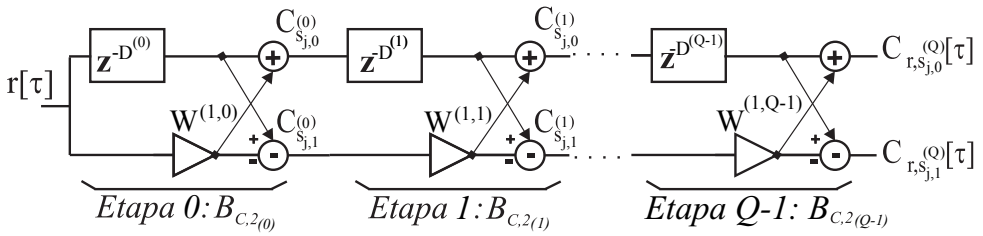


Figura 2.4: Diagrama de bloques del correlador eficiente Golay (EGC) [Bud91, Pop99].

El algoritmo eficiente de generación y correlación de parejas Golay fue posteriormente am-

pliado al caso de 4-CSS (conjuntos de cuatro secuencias complementarias) por F. J. Álvarez *et al.* [ÁUM+04] y más tarde a K-CSS por C. De Marziani *et al.* [MUH+07], generando cualquier número $K = 2^k$ de secuencias complementarias de longitud $L = K^Q$.

También, en 1990 Budisin propone un algoritmo para la generación y correlación de pares de secuencias complementarias multinivel, PMCS (*Pairs of Multilevel Complementary Sequences*) [Bud90], pero sólo se limita a generar dos conjuntos mutuamente incorrelados.

Recientemente, E. Garcia [GUG13] propuso dos arquitecturas modulares para la generación y correlación de K conjuntos complementarios multinivel utilizando algoritmos recursivos: un algoritmo para $K = 2^k$, ($k \in \mathbb{N} - \{0\}$) CSS multinivel, y otro para $K \geq 3 - \{4\}$ CSS multinivel. Por tanto, con ambos algoritmos puede generar y correlar eficientemente K conjuntos CSS multinivel, con $K \in \mathbb{N} - \{0, 1\}$. En este trabajo se llevará a cabo la implementación hardware del primero de ellos, quedando como trabajo futuro la implementación del segundo.

2.4. Plataformas para implementación Hardware

A la hora de implementar el hardware, existen varias alternativas tecnológicas para la implementación del sistema bajo análisis. A continuación se detallarán las características principales de las tecnologías más utilizadas actualmente:

2.4.1. Microcontrolador

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento (CPU), memorias que suelen ser tipo ROM/PROM/EPROM/EEPROM/FLASH donde se almacenan los datos del programa de control, y periféricos de entrada/salida, como temporizadores, puertos serie/paralelo, convertidores A/D y D/A, etc. Están diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Sin embargo, debido a su carácter genérico, no optimiza recursos ni tiempos de operación, por lo que carece de la eficiencia de otros dispositivos.

2.4.2. DSPs (*Digital Signal Processor*)

Un DSP es un sistema basado en un microprocesador que posee un conjunto de instrucciones, un hardware y un software optimizados para el procesado de señales digitales. Para poder realizar estas operaciones de procesado como son el filtrado o la FFT, es preciso realizar operaciones aritméticas complejas y repetitivas, trabajando en tiempo real. Por esta razón, la arquitectura de los DSPs está pensada para realizar de forma eficiente el mayor número de operaciones posible.

Estos dispositivos suelen estar basados en la arquitectura Harvard, la cual consiste en separar la memoria interna de programa de la de datos. De esta manera es posible acceder de forma simultánea a través de dos buses diferentes, tanto a los datos como a las instrucciones consiguiendo una mayor velocidad en la ejecución del programa.

La CPU está formada por varios registros de uso general así como una serie de ALUs y multiplicadores, que permiten llevar a cabo operaciones aritméticas y lógicas en paralelo. Para realizar estas operaciones, la CPU cuenta con una serie de unidades funcionales. Otra de las características de la arquitectura de los DSPs es el uso de VLIW (*Very Long Instruction Word*). Basándose en que la CPU consta de varias ALUs y multiplicadores, se realiza una lectura de varias instrucciones cada vez que se lleva a cabo un acceso a memoria de programa, permitiendo

la ejecución de varias instrucciones de forma simultánea. El tamaño de esta palabra es variable y la ejecución de las instrucciones en paralelo depende de los recursos que se estén utilizando en cada momento.

Otro elemento que suele ser común son los bloques DMA (*Direct Memory Access*), que permiten el acceso a memoria externa y a los periféricos de forma eficiente. Puesto que la mayor parte de las operaciones relacionadas con el procesado de la señal en tiempo discreto, como la convolución o la transformada de Fourier, se realizan mediante la implementación de suma de productos, estos dispositivos vienen preparados para realizar estas operaciones específicas. Cuando las frecuencias de muestreo crecen por encima de unos pocos MHz, un DSP tiene que trabajar mucho para transferir los datos sin ninguna pérdida. Esto es porque el procesador debe utilizar los recursos compartidos como buses de memoria, o incluso el núcleo del procesador para otras tareas como interrupciones y pueden llegar a colapsarse por algún tiempo.

El DSP se puede programar tanto en lenguaje ensamblador como en C. Este código C puede tener un alto nivel de ramificación y de toma de decisiones, como por ejemplo, las pilas de protocolos de sistemas de comunicaciones.

Un DSP está diseñado para ofrecer el re-uso de las unidades de procesamiento, por ejemplo, un multiplicador que se utiliza para el cálculo de una FIR puede ser re-utilizado por otra rutina que calcula la FFT. Es por esto que si se requiere un cambio de contexto importante, el DSP puede implementarlo por bifurcación a una parte nueva del programa.

2.4.3. ASIC (*Application-Specific Integrated Circuit*)

Los circuitos integrados ASIC son circuitos diseñados para una aplicación específica, es decir, los ASIC son fabricados con toda la funcionalidad requerida para un determinado diseño, por lo que una de sus desventajas es que no puede reutilizarse para otra función que no sea para la cual se diseñó. Estos dispositivos pueden contener funciones analógicas, digitales y combinaciones de ambas. Son dispositivos con altas prestaciones pero sólo resultan rentables para fabricaciones masivas ya que sus costes de diseño son muy elevados.

Según el método seguido en su diseño se tienen:

- *Full-Custom*: El diseño se lleva a cabo totalmente a medida. Se consiguen áreas reducidas (y consecuentemente costos por unidad menores), un funcionamiento optimizado, y la posibilidad de integrar componentes analógicos. No obstante, presenta como inconvenientes un costo y tiempo de desarrollo elevados, mayor complejidad en las herramientas de desarrollo, así como la necesidad de diseñadores expertos.
- *Semi-Custom*: El dispositivo se diseña a partir del interconexionado entre bloques funcionales existentes. De esta manera se logra el abaratamiento de costes y la reducción de los tiempos de desarrollo. Aunque el área que ocupan es mayor y nunca se aprovecha el 100 % de los dispositivos interconectados.

2.4.4. FPGA (*Field-Programmable Gate Array*)

La FPGA es un dispositivo constituido por bloques lógicos cuya interconexión y funcionalidad puede ser configurada por el usuario mediante un lenguaje de descripción de hardware (VHDL por ejemplo), lo cual simplifica de manera considerable la complejidad de los diseños. Los bloques lógicos presentan una funcionalidad reducida por sí solos, pero gracias a la gran cantidad de éstos, es posible implementar funcionalidades complejas.

Las FPGAs se utilizan en aplicaciones similares a los ASICs, sin embargo son más lentas, consumen más potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de

esto, las FPGAs tienen las ventajas de ser reprogramables, por lo que sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos. El tiempo de desarrollo es también menor, ya que la etapa de pruebas y testado son menos costosas y complejas que con los dispositivos ASICs.

Las FPGAs fueron introducidas por Xilinx en 1985 y fueron creadas para proporcionar una solución intermedia entre los ASICs y los CPLDs (*Complex Programmable Logic Device*). Las principales ventajas de las FPGAs frente a los ASICs son el tiempo, y por lo tanto el coste, de llevar a cabo un desarrollo.

Las FPGAs presentan tres tipos de elementos programables:

1. Bloques lógicos (CLB).
2. Bloques de entrada salida (IOB).
3. Matrices de interconexión (SM).

En la Figura 2.5, se puede observar la interconexión entre los distintos elementos programables de la FPGA, en su estructura interna.

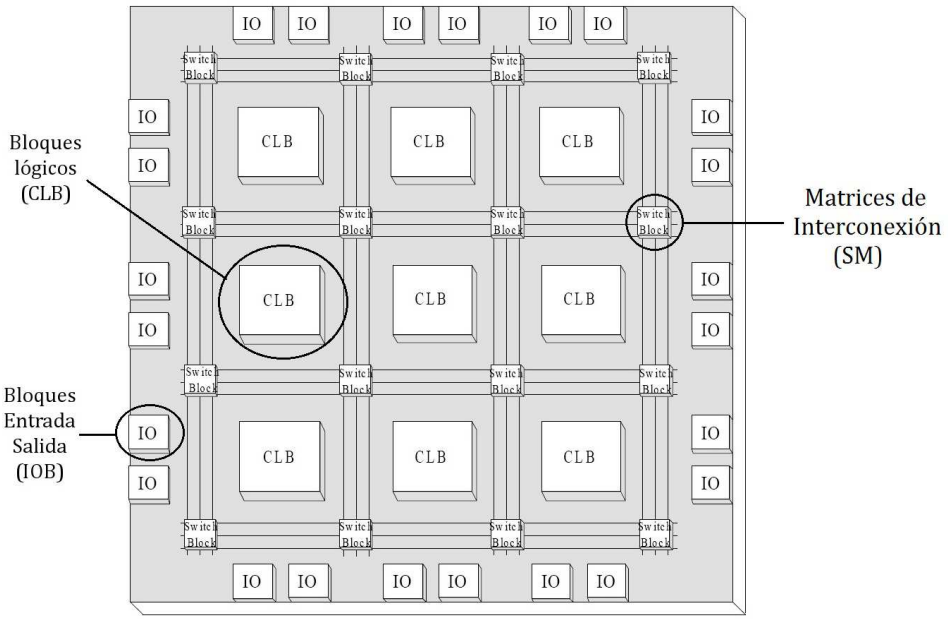


Figura 2.5: Esquema de la estructura interna de una FPGA, con sus tres partes principales [EDA11].

CLB (*Configurable Logic Block*): cada uno de estos bloques está formado por elementos de lógica combinacional denominados LUT (*Look Up Table*) y un registro que puede ser configurado en modo *latch* o *flip-flop*. La lógica combinacional permite implementar funciones *booleanas* a partir de las entradas. La salida de las LUTs puede ir a un registro cuya salida va a un multiplexor, o bien conectarla directamente al multiplexor sin pasar por el registro. De esta manera es posible configurar las salidas del CLB como secuenciales o combinacionales. En la Figura 2.6 se muestra la estructura típica de un CLB.

IOB (*Input-Ouput Block*): estos bloques se sitúan en los extremos de la FPGA, y como su nombre indica, son bloques de entrada/salida. Cada bloque puede ser configurado de forma independiente para funcionar como entrada, salida o bidireccional, pudiendo configurarse además, como tri-estado. Los IOBs pueden programarse para trabajar con diferentes niveles de tensión y cada uno incluye (*flip-flops*) que pueden utilizarse para registrar tanto las entradas como las salidas. En la Figura 2.7 se muestra la arquitectura típica de un IOB.

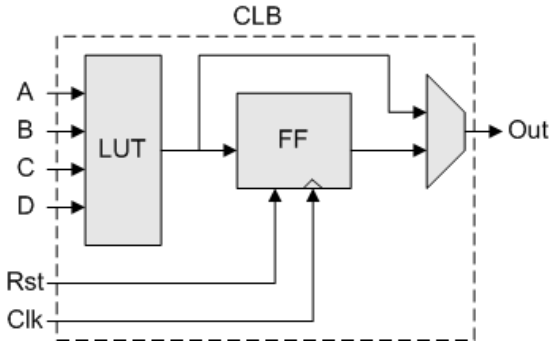


Figura 2.6: Esquema del bloque CLB de una FPGA [Wik07].

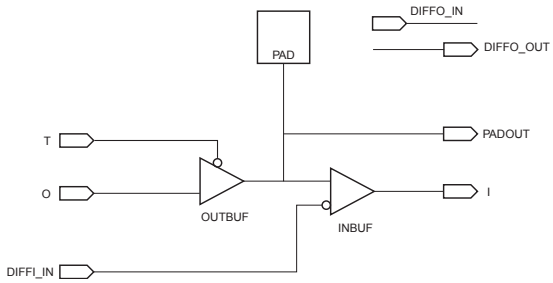


Figura 2.7: Esquema del bloque IOB de la FPGA [Xil08].

Líneas de interconexión: constituyen los caminos que permiten interconectar las entradas y salidas de los diferentes bloques. Están formadas por líneas de dos capas que recorren horizontal y verticalmente las filas y columnas existentes entre los CLBs. Dos elementos adicionales participan activamente en el proceso de conexión:

- Puntos de Interconexión Programable (PIP, *Programmable Interconnection Point*): permiten la conexión de CLBs e IOBs a líneas cercanas.
- Matrices de interconexión (SW, *Switch Matrix*): son dispositivos de conmutación distribuidos de forma uniforme por la FPGA y permite conectar señales de unas líneas a otras.

2.4.5. Discusión

Si se tuvieran que comparar estas plataformas, para implementar un correlador eficiente de secuencias complementarias multinivel, los microcontroladores quedarían descartados en primer lugar, ya que no se pueden utilizar para realizar procesamiento en paralelo, aunque si pueden realizar operaciones de suma, resta y multiplicación, su uso está más orientado al procesamiento secuencial. El DSP por otro lado, es más eficiente para realizar operaciones matemáticas ya que posee un microprocesador con un conjunto de instrucciones, y tiene implementadas las funciones aritméticas como hardware y tiene varios módulos de multiplicación y suma que consumen poca energía, lo que permite paralelizar procesos. Sin embargo, los recursos que ofrecen los DSPs son limitados y escasos, por lo que si se quiere implementar un correlador con varias etapas los recursos no alcanzarían. La ventaja de una FPGA frente al DSP, es que contiene mayor cantidad de módulos dedicados de multiplicación, y la arquitectura se puede realizar de una forma modular, por lo que agregar más etapas a una determinada implementación, simplemente sería replicar un bloque dentro de la FPGA hasta el límite de recursos según el modelo utilizado. En cuanto a consumo de energía, las FPGAs consumen más que los DSPs, pero permiten paralelizar mayor cantidad de procesos, por lo que son mas eficientes que los DSPs. En cuanto a los ASICs, pueden implementar cualquier función de manera óptima y con muy bajo consumo de energía,

sin embargo, el hardware no es reconfigurable y los costos de producción y diseño son muy elevados.

Desde los primeros trabajos de procesamiento con correladores digitales se intuye la necesidad de definir el mejor soporte para la implementación, aprovechando su naturaleza y buscando su ejecución en tiempo real, sin que ello suponga un coste elevado. Las arquitecturas basadas en FPGA ya han sido utilizadas en trabajos previos con éxito, como la implementación de un correlador eficiente de pares Golay (EGC) aplicado para sistemas ultrasónicos [HUH+03] o la implementación hardware de correladores eficientes para 4-CSS [ÁHU+06] o K-CSS [PUH+06]. El uso de estas arquitecturas está especialmente indicado para la paralelización de procesos y el manejo de gran volumen de datos. Por lo que, una vez revisado las diferentes plataformas existentes para la implementación de algoritmos de procesado de señal y considerando los trabajos previos que han implementado arquitecturas de correlación para CSS, en este trabajo se plantea la implementación hardware de un correlador eficiente de conjuntos de secuencias complementarias multinivel, desarrollado en forma teórica por E. García en [GUG13], sobre una plataforma FPGA de Xilinx.

Capítulo 3

Conjunto de Secuencias Complementarias Multinivel

Tal como se mencionó anteriormente, en [GUG13] E. García *et al.* proponen dos generalizaciones de arquitecturas eficientes para la generación y correlación de $K_{|MultCSS}$ conjuntos CSS multinivel: la primera para $K_{|MultCSS} = 2^k,\ k \in \mathbb{N} - \{0\}$ CSS multinivel, y la segunda para $K_{|MultCSS} \geq 3 - \{4\}$ CSS multinivel. Por consiguiente, con ambos algoritmos es posible generar y/o correlar eficientemente $K_{|MultCSS}$ CSS multinivel, con $K_{|MultCSS} \geq 2$, y sus longitudes pueden ser ajustadas al valor deseado sin restricciones. Lo interesante de estas arquitecturas, es que además pueden ser la base para la generación de CSS multinivel de longitudes flexibles, o para generar pares ternarios óptimos [GUG13]. En este trabajo, solo se tendrá en cuenta el primer algoritmo para $K_{|MultCSS} = 2^k$, el cual será implementado en hardware.

3.1. Generalización del algoritmo eficiente para CSS al alfabeto multinivel

En 1995 Wornell [Wor95] propuso un algoritmo eficiente para la generación de K conjuntos complementarios de longitud $L = K^Q$ que es definido como:

$$\begin{aligned} \mathbf{S}_j^{(k,0)}[l] &= \boldsymbol{\mathcal{H}}_K \\ \mathbf{S}_j^{(k,q+1)}[l] &= \boldsymbol{\mathcal{H}}_K \times \mathbf{D}^{(k,q)}[l] \times \mathbf{S}_j^{(k,q)}[l] \end{aligned} \tag{3.1}$$

donde se definen los siguientes parámetros:

- $\boldsymbol{\mathcal{H}}_K$ es una matriz de Hadamard[1]generada con el método recursivo de Sylvester [Syl67], para $K = 2^k$.
- k es el número de iteraciones del algoritmo de Sylvester necesarias para la generación de la matriz Hadamard $\boldsymbol{\mathcal{H}}_K, k = log_2(K), k \geq 1$. También es llamado factor de expansión.
- $\mathbf{S}_j^{(k,q+1)}[l]$ es el j-ésimo conjunto CSS de K secuencias de longitud L tras la iteración q. $\mathbf{S}_j^{(k,q+1)}[l] = [s_{j,0}^{(q+1)}[l]\ \ldots\ s_{j,K-1}^{(q+1)}[l]]^T$.
- q es el número de iteraciones del algoritmo de Wornell, $0 \leq q \leq Q-1, Q \in \mathbb{N} - \{0\}$
- $\mathbf{D}^{(k,q)}$ es una matriz de retardos definida como:

[1]Una matriz $\boldsymbol{\mathcal{H}}_K$ de dimensiones $K \times K$ con entradas ± 1 es una matriz Hadamard si $\boldsymbol{\mathcal{H}}_K^T \times \boldsymbol{\mathcal{H}}_K = K \cdot \mathbf{I}_K$, donde $\mathbf{I}_K$ es la matriz identidad.

$$\mathbf{D}^{(k,q)}[l] = \begin{bmatrix} \delta[l] & 0 & \cdots & 0 \\ 0 & \delta[l-K^q] & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \delta[l-(K-1).K^q] \end{bmatrix} \tag{3.2}$$

Cabe mencionar, que este algoritmo es una generalización del algoritmo conocido como Golay-Rudin-Shapiro [Sha51, Rud59], y solo puede generar un CSS de K secuencias.

En [ÁUM+04] F. J. Álvarez *et al.*, generalizan el algoritmo de Wornell para $K = 4$ CSS binarios incorrelados, y posteriormente en [MUH+07] De Marziani *et al.* proponen la generalización para $K = 2^k$ CSS binarios incorrelados.

Esta última generalización del algoritmo de Wornell se puede expresar en el dominio de tiempo discreto como:

$$\begin{aligned} \mathbf{S}_j^{(k,0)}[l] &= \delta[l] \cdot [1 \ \cdots \ 1]_{1,K}^T \\ \mathbf{S}_j^{(k,q+1)}[l] &= \mathbf{\Lambda}_j^{(k,q)}[l] \times \mathbf{D}^{(k,q)}[l] \times \mathbf{S}_j^{(k,q)}[l] \end{aligned} \tag{3.3}$$

donde la matriz $\mathbf{\Lambda}_j^{(k,q)}$ es una matriz Hadamard de orden $K = 2^k, k \in \mathbb{N} - \{0\}$, generada con el método de entrelazado de Tseng-Liu [TL72]; el término $\mathbf{S}_j^{(k,q+1)}[l]$ denota el j-ésimo conjunto complementario ($0 \leq j \leq K-1$), compuesto por K secuencias binarias después de la iteración q ($0 \leq q \leq Q-1$), es decir $\mathbf{S}_j^{(k,q+1)}[l] = [s_{j,0}^{(q+1)}[l] \ \ldots \ s_{j,K-1}^{(q+1)}[l]]^T$. La matriz $\mathbf{D}^{(k,q)}[l]$ es una matriz diagonal definida según (3.4), donde los elementos de retardo $D^{(q)}$ son elegidos como una permutación de los valores $\{K^0, K^1, \ldots, K^{Q-1}\}$.

$$\mathbf{D}^{(k,q)}[l] = \begin{bmatrix} \delta[l] & 0 & \cdots & 0 \\ 0 & \delta[l-D^{(q)}] & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \delta[l-(K-1)\cdot D^{(q)}] \end{bmatrix} \tag{3.4}$$

Las matrices de Hadamard $\mathbf{\Lambda}_j^{(k,q)}$ de orden K se generan a partir del método de entrelazado o concatenación de Tseng-Liu para un conjunto complementario dado $\mathbf{S}_j^{(k,Q)}[l]$ [TL72]. En adelante, a la operación de entrelazado se la define con el símbolo $\odot$.

Este método que empieza para $k = 2$, genera en cada paso k de expansión dos nuevas matrices, cada una de ellas representa un conjunto CSS de 2^k secuencias y de longitud 2^k, a su vez, ambos CSS son incorrelados entre sí. El algoritmo anterior se unifica en [MUH+07] como:

$$\mathbf{\Lambda}_j^{(k,q)} = \begin{bmatrix} \mathbf{\Lambda}_j^{(k-1,q)} \odot \left(-W^{(k,q)} \cdot \mathbf{\Lambda}_j^{(k-1,q)}\right) \\ \mathbf{\Lambda}_j^{(k-1,q)} \odot \left(W^{(k,q)} \cdot \mathbf{\Lambda}_j^{(k-1,q)}\right) \end{bmatrix} \tag{3.5}$$

Con el fin de generar K CSS incorrelados de longitud $L = K^Q$, se tienen que usar todas las variaciones con repetición de las semillas $W^{(k,0)} \in \{-1, +1\}$ en la primera iteración $q = 0$, tomados de k en k. El valor de las semillas para las demás iteraciones, $\left\{W^{(k,1)}, \ldots, W^{(k,Q-1)}\right\}$ deben ser igual para todos los conjuntos.

Se pretende por tanto generalizar el algoritmo anterior al alfabeto multinivel. En [Bud90] Budisin propone un algoritmo eficiente para generar un par de secuencias complementarias multinivel ($K_{|MultCSS} = 2$), expresado según (3.6):

$$\begin{aligned} s_{j,0}^{(0)}[l] &= \delta[l] \\ s_{j,1}^{(0)}[l] &= A^{(0)} \cdot \delta[l] \\ s_{j,0}^{(q+1)}[l] &= s_{j,0}^{(q)}[l] + A^{(q)} \cdot s_{j,1}^{(q)}[l - D^{(q)}] \\ s_{j,1}^{(q+1)}[l] &= A^{(q)} \cdot s_{j,0}^{(q)}[l] - s_{j,1}^{(q)}[l - D^{(q)}] \end{aligned} \tag{3.6}$$

donde se definen los siguientes parámetros:

- $s_{j,i}^{(q+1)}[l]$ es la secuencia i-ésima del conjunto j-ésimo, $(i, j \in \{0, \ldots, K_{|MultCSS} - 1\})$ luego de la iteración q.
- q es el número de iteración, $q \in \{0, \ldots, Q-1\}$ y $Q \in \mathbb{N}$ es el número total de iteraciones del algoritmo.
- $A^{(q)}$ es el valor del multiplicador en la iteración q, éste puede tomar cualquier valor real.

Con el fin de lograr la generalización del algoritmo eficiente propuesto por De Marziani *et al.* al alfabeto multinivel, se modifica el algoritmo de generación (3.6) y se expresa en forma matricial según la expresión (3.7), para obtener el equivalente de la siguiente manera:

$$\begin{bmatrix} s_{j,0}^{(q+1)}[l] \\ s_{j,1}^{(q+1)}[l] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & A^{(q)} \cdot W^{(1,q)} \\ A^{(q)} & -W^{(1,q)} \end{bmatrix}}_{\mathbf{\Lambda}_j^{(1,q)}} \times \underbrace{\begin{bmatrix} \delta[l] & 0 \\ 0 & \delta[l - D^{(q)}] \end{bmatrix}}_{\mathbf{D}^{(1,q)}[l]} \times \begin{bmatrix} s_{j,0}^{(q)}[l] \\ s_{j,1}^{(q)}[l] \end{bmatrix} \tag{3.7}$$

De esta forma se redefine el término $\mathbf{S}_j^{(k,q+1)}[l]$ como el j-ésimo conjunto complementario multinivel $(0 \leq j \leq K_{|MultCSS} - 1)$, compuesto por $K_{|MultCSS} = 2^k$ $(k \in \mathbb{N} - \{0\})$ secuencias multinivel luego de la iteración q $(0 \leq q \leq Q - 1)$.

Es decir, en términos sencillos, que a partir de una $\delta[l]$ como parámetro inicial, y después de sucesivos retardos, sumas y multiplicaciones, se van formando secuencias complementarias a lo largo de cada iteración q, que se van entrelazando según la expresión (3.8) y formando secuencias complementarias de mayor longitud, hasta obtener el j-ésimo conjunto complementario multinivel $\mathbf{S}_j^{(k,q+1)}[l]$, dado a partir de una semilla de generación específica. Para generar los demás conjuntos complementarios incorrelados basta con utilizar todas las variaciones con repetición de las semillas $W^{(k,0)}$ de la etapa $q = 0$, y las demás mantenerlas fijas.

Para entender un poco mejor el funcionamiento del algoritmo, se considerará como ejemplo la generación de $K_{|MultCSS} = 2^2 = 4$ secuencias (ej. $k = 2$) de longitud $L_{|MultCSS} = 4^2 = 16$ con el algoritmo generalizado en [MUH+07]. El número de etapas de este algoritmo es igual a $Q = 2$, los retardos $D^{(q)}$ son elegidos como $D^{(q)} = \{4^1, 4^0\}$ y los valores de $A^{(q)} = \{1, 3\}$. Utilizando la ecuación recursiva (3.5), la matriz $\mathbf{\Lambda}_j^{(2,q)}$ resulta de la siguiente manera:

$$\begin{aligned} \mathbf{\Lambda}_j^{(2,q)} &= \begin{bmatrix} \mathbf{\Lambda}_j^{(1,q)} \odot \left(-W^{(2,q)} \cdot \mathbf{\Lambda}_j^{(1,q)}\right) \\ \mathbf{\Lambda}_j^{(1,q)} \odot \left(W^{(2,q)} \cdot \mathbf{\Lambda}_j^{(1,q)}\right) \end{bmatrix} \\ &= \begin{bmatrix} 1 & -W^{(2,q)} & W^{(1,q)} \cdot A^{(q)} & -W^{(1,q)} \cdot W^{(2,q)} \cdot A^{(q)} \\ A^{(q)} & -W^{(2,q)} \cdot A^{(q)} & -W^{(1,q)} & W^{(1,q)} \cdot W^{(2,q)} \\ 1 & W^{(2,q)} & W^{(1,q)} \cdot A^{(q)} & W^{(1,q)} \cdot W^{(2,q)} \cdot A^{(q)} \\ A^{(q)} & W^{(2,q)} \cdot A^{(q)} & -W^{(1,q)} & -W^{(1,q)} \cdot W^{(2,q)} \end{bmatrix} \end{aligned} \tag{3.8}$$

A partir de la matriz de Hadamard de la expresión anterior y la generalización de (3.7), se obtienen las ecuaciones que expresan las 4 secuencias generadas como:

$$s_{j,0}^{(0)}[l] = s_{j,1}^{(0)}[l] = s_{j,2}^{(0)}[l] = s_{j,3}^{(0)}[l] = \delta[l]$$

$$\begin{aligned}
s_{j,0}^{(q+1)}[l] &= s_{j,0}^{(q)}[l] - W^{(2,q)} \cdot s_{j,1}^{(q)}[l - D^{(q)}] + W^{(1,q)} \cdot A^{(q)} \cdot s_{j,2}^{(q)}[l - 2 \cdot D^{(q)}] \\
&\quad - A^{(q)} \cdot W^{(1.q)} \cdot W^{(2,q)} \cdot s_{j,3}^{(q)}[l - 3 \cdot D^{(q)}] \\
s_{j,1}^{(q+1)}[l] &= A^{(q)} \cdot s_{j,0}^{(q)}[l] - W^{(2,q)} \cdot A^{(q)} \cdot s_{j,1}^{(q)}[l - D^{(q)}] - W^{(1,q)} \cdot s_{j,2}^{(q)}[l - 2 \cdot D^{(q)}] \\
&\quad + W^{(1.q)} \cdot W^{(2,q)} \cdot s_{j,3}^{(q)}[l - 3 \cdot D^{(q)}] \\
s_{j,2}^{(q+1)}[l] &= s_{j,0}^{(q)}[l] + W^{(2,q)} \cdot s_{j,1}^{(q)}[l - D^{(q)}] + W^{(1,q)} \cdot A^{(q)} \cdot s_{j,2}^{(q)}[l - 2 \cdot D^{(q)}] \\
&\quad + A^{(q)} \cdot W^{(1.q)} \cdot W^{(2,q)} \cdot s_{j,3}^{(q)}[l - 3 \cdot D^{(q)}] \\
s_{j,3}^{(q+1)}[l] &= A^{(q)} \cdot s_{j,0}^{(q)}[l] + W^{(2,q)} \cdot A^{(q)} \cdot s_{j,1}^{(q)}[l - D^{(q)}] - W^{(1,q)} \cdot s_{j,2}^{(q)}[l - 2 \cdot D^{(q)}] \\
&\quad - W^{(1.q)} \cdot W^{(2,q)} \cdot s_{j,3}^{(q)}[l - 3 \cdot D^{(q)}]
\end{aligned} \tag{3.9}$$

En la Tabla 3.1 se observan las $K_{|MultCSS} = 4$ variaciones con repetición de los valores de la semilla $W^{(1,0)}, W^{(2,0)}$ y las matrices resultantes de $\mathbf{\Lambda}_j^{(2,0)}$.

$\mathbf{\Lambda}_0^{(2,0)}$; $W^{(1,0)} = +1, W^{(2,0)} = +1$	$\mathbf{\Lambda}_1^{(2,0)}$; $W^{(1,0)} = -1, W^{(2,0)} = +1$
$\begin{bmatrix} 1 & -1 & A^{(0)} & -A^{(0)} \\ A^{(0)} & -A^{(0)} & -1 & 1 \\ 1 & 1 & A^{(0)} & A^{(0)} \\ A^{(0)} & A^{(0)} & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 & -A^{(0)} & A^{(0)} \\ A^{(0)} & -A^{(0)} & 1 & -1 \\ 1 & 1 & A^{(0)} & -A^{(0)} \\ A^{(0)} & A^{(0)} & -1 & 1 \end{bmatrix}$
$\mathbf{\Lambda}_2^{(2,0)}$; $W^{(1,0)} = +1, W^{(2,0)} = -1$	$\mathbf{\Lambda}_3^{(2,0)}$; $W^{(1,0)} = -1, W^{(2,0)} = -1$
$\begin{bmatrix} 1 & 1 & A^{(0)} & A^{(0)} \\ A^{(0)} & A^{(0)} & -1 & -1 \\ 1 & -1 & A^{(0)} & -A^{(0)} \\ A^{(0)} & -A^{(0)} & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & -A^{(0)} & -A^{(0)} \\ A^{(0)} & A^{(0)} & 1 & 1 \\ 1 & -1 & -A^{(0)} & A^{(0)} \\ A^{(0)} & -A^{(0)} & 1 & -1 \end{bmatrix}$

Tabla 3.1: Matrices $\mathbf{\Lambda}_j^{(2,0)}$ para la generación de 4 CSS obtenidos como la variación con repetición de los valores de la semilla en la primera etapa [GUG13]. Cabe mencionar que si los valores de los multiplicadores fueran $A^{(q)} = \{1, 1\}$, entonces esta matriz generaría secuencias binarias.

Si se considera la generación para el caso del conjunto complementario multinivel correspondiente a la variación de la semilla $W^{(1,0)} = -1$ y $W^{(2,0)} = -1$, y con los valores para la etapa $q = 1$ elegidos como $W^{(1,1)} = -1$ y $W^{(2,1)} = -1$, se obtiene la generación de 4-CSS multinivel pasadas 2 iteraciones, de la siguiente manera:

Partiendo de una $\delta[l]$, como condición inicial se tiene:

$$\begin{aligned}
s_{3,0}^{(0)}[l] &= \{+1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\} \\
s_{3,\ 1}^{(0)}[l] &= \{+1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\} \\
s_{3,\ 2}^{(0)}[l] &= \{+1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\} \\
s_{3,\ 3}^{(0)}[l] &= \{+1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,0,\ 0\}
\end{aligned} \tag{3.10}$$

A continuación, en la iteración $q = 1$, siguiendo la expresión (3.9), aplicando los retardos correspondientes y la matriz de Hadamard, se obtiene:

$$
\begin{aligned}
s^{(1)}_{3,0}[l] &= s^{(0)}_{3,0}[l] + s^{(0)}_{3,1}[l-4] - s^{(0)}_{3,2}[l-8] - s^{(0)}_{3,3}[l-12] \\
s^{(1)}_{3,1}[l] &= s^{(0)}_{3,0}[l] + s^{(0)}_{3,1}[l-4] + s^{(0)}_{3,2}[l-8] + s^{(0)}_{3,3}[l-12] \\
s^{(1)}_{3,2}[l] &= s^{(0)}_{3,0}[l] - s^{(0)}_{3,1}[l-4] - s^{(0)}_{3,2}[l-8] + s^{(0)}_{3,3}[l-12] \\
s^{(1)}_{3,3}[l] &= s^{(0)}_{3,0}[l] - s^{(0)}_{3,1}[l-4] + s^{(0)}_{3,2}[l-8] - s^{(0)}_{3,3}[l-12]
\end{aligned}
$$

$$
\begin{aligned}
s^{(0)}_{3,0}[l] &= \{+1,\ 0,\ 0,\ 0,\ +1,\ 0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0\} \\
s^{(0)}_{3,1}[l] &= \{+1,\ 0,\ 0,\ 0,\ +1,\ 0,\ 0,\ 0,\ +1,\ 0,\ 0,\ 0,\ +1,\ 0,\ 0,\ 0\} \\
s^{(0)}_{3,2}[l] &= \{+1,\ 0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0,\ +1,\ 0,\ 0,\ 0\} \\
s^{(0)}_{3,3}[l] &= \{+1,\ 0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0,\ +1,\ 0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0\}
\end{aligned}
$$

Y por último en la iteración 2, se obtiene la expresión final

$$
\begin{aligned}
s^{(1)}_{3,0}[l] &= s^{(1)}_{3,0}[l] + s^{(1)}_{3,1}[l-1] - 3 \cdot s^{(1)}_{3,2}[l-2] - 3 \cdot s^{(1)}_{3,3}[l-3] \\
s^{(1)}_{3,1}[l] &= 3 \cdot s^{(1)}_{3,0}[l] + 3 \cdot s^{(1)}_{3,1}[l-1] + s^{(1)}_{3,2}[l-2] + s^{(1)}_{3,3}[l-3] \\
s^{(1)}_{3,2}[l] &= s^{(1)}_{3,0}[l] - s^{(1)}_{3,1}[l-1] - 3 \cdot s^{(1)}_{3,2}[l-2] + 3 \cdot s^{(1)}_{3,3}[l-3] \\
s^{(1)}_{3,3}[l] &= 3 \cdot s^{(1)}_{3,0}[l] - 3 \cdot s^{(1)}_{3,1}[l-1] + s^{(1)}_{3,2}[l-2] - s^{(1)}_{3,3}[l-3]
\end{aligned}
$$

Y el conjunto de secuencias generadas, para la semilla utilizada, es el siguiente:

$$
\begin{aligned}
s^{(2)}_{3,0}[l] &= \{+1,+1,-3,-3,+1,+1,+3,+3,-1,+1,+3,-3,-1,+1,-3,+3\} \\
s^{(2)}_{3,1}[l] &= \{+3,+3,+1,+1,+3,+3,-1,-1,-3,+3,-1,+1,-3,+3,+1,-1\} \\
s^{(2)}_{3,2}[l] &= \{+1,-1,-3,+3,+1,-1,+3,-3,-1,-1,+3,+3,-1,-1,-3,-3\} \\
s^{(2)}_{3,3}[l] &= \{+3,-3,+1,-1,+3,-3,-1,+1,-3,-3,-1,-1,-3,-3,+1,+1\} \qquad (3.11)
\end{aligned}
$$

3.2. Correlación de CSS multinivel

3.2.1. Algoritmos eficientes de correlación para CSS binarias

En la sección 2.3, se mencionó el trabajo realizado por C. De Marziani *et al.* [MUH+07], en el que desarrollan un algoritmo eficiente para la generación y correlación de conjuntos CSS binarios, posteriormente M. C. Pérez *et al.* en [PUH+07] implementan dicho algoritmo en una plataforma FPGA. Para generalizar el algoritmo CSS binario, implementado en [PUH+07], y trasladarlo al alfabeto multinivel según lo propuesto en [GUG13], se deberán plantear algunas modificaciones a dicha implementación para alcanzar el objetivo enunciado.

Por ello, se describirán a continuación los principios básicos del algoritmo eficiente para CSS binario, con objeto de comprender sus características principales.

En [MUH+07] se ofrece un algoritmo generalizado que incluye los propuestos para parejas Golay (2-CSS) en [Pop99, Bud91], cuya arquitectura se ve en la Figura 3.1, y para conjuntos

de cuatro secuencias complementarias (4-CSS), como el desarrollado en [ÁUM+04] (ver Figura 3.2). Este algoritmo generalizado, permite la construcción de generadores y correladores eficientes de conjuntos de cualquier número $K = 2^k$ de secuencias complementarias de longitud $L = K^Q$. Sin entrar en detalle, se puede observar en la Figura 3.3 el diagrama en bloques del generador eficiente de CSS, ESSG (*Efficient Set of Sequences Generator*), y considerarlo como un filtro digital compuesto por Q etapas, a su vez cada una de ellas contiene elementos de retardo, sumadores, restadores y multiplicadores. Para que las secuencias generadas sean binarias, con valores 1 y -1, los retardos $D^{(q)}$ deben escogerse como cualquier permutación del conjunto $\{K^0, K^1, K^2, \ldots, K^{Q-1}\}$. Es más, si se desea evitar el desbordamiento tras las operaciones internas de cada etapa, se necesitarían $k = log_2(K)$ bits adicionales de memoria en cada etapa para cada muestra almacenada. A su vez, si se elige la permutación inversa $\{K^{Q-1}, K^{Q-2}, \ldots, K^0\}$ para los retardos $D^{(q)}$, de modo que a la primera etapa le correspondan los retardos de mayor valor, se puede reducir la memoria total necesaria para la implementación del generador.

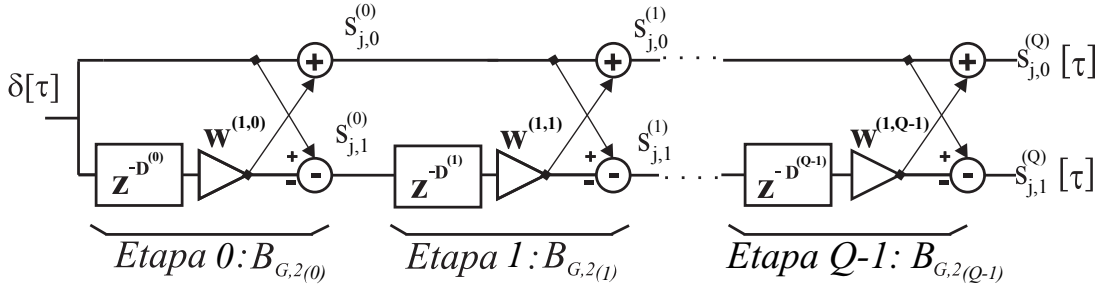


Figura 3.1: Diagrama de bloques del generador eficiente de 2-CSS (2-ESSG) propuesto por [Bud91, Pop99].

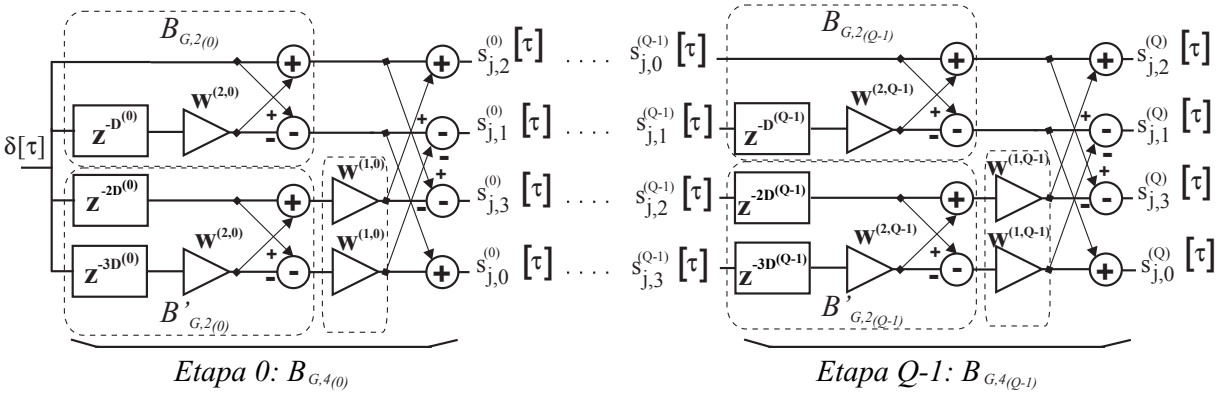


Figura 3.2: Diagrama de bloques del generador eficiente de 4-CSS (4-ESSG) propuesto por [ÁUM+04].

Los coeficientes $W^{(k,q)} = (W^{(1,0)}, \ldots, W^{(k,0)}, W^{(1,1)}, \ldots, W^{(k,1)}, \ldots, W^{(1,Q-1)}, \ldots, W^{(k,Q-1)})$ constituyen la semilla de generación del conjunto, pudiendo tomar valores 1 o -1. De esta manera, las operaciones de multiplicación se reducen a sumas y restas por estos coeficientes. Teniendo en cuenta este esquema de generación, es muy sencillo obtener K CSS mutuamente incorrelados: solo hay que realizar todas las combinaciones posibles de los coeficientes $(W^{(1,0)}, W^{(2,0)}, \ldots, W^{(k,0)})$ de la primera etapa y fijar el resto de los coeficientes a un valor determinado. Por otro lado, y para mayor comodidad, el vector con los coeficientes $W^{(k,q)}$ se puede representar en base decimal como $W_d^{(k,q)}$ considerando los valores -1 como 0 y siendo $W^{(1,0)}$ el bit más significativo. Se puede observar también en la Figura 3.3 que el esquema del generador posee K salidas de las cuales se obtienen las secuencias complementarias cuando la entrada es un impulso $\delta[\tau]$.

Sabiendo que la respuesta al impulso de un filtro a una secuencia determinada es la versión

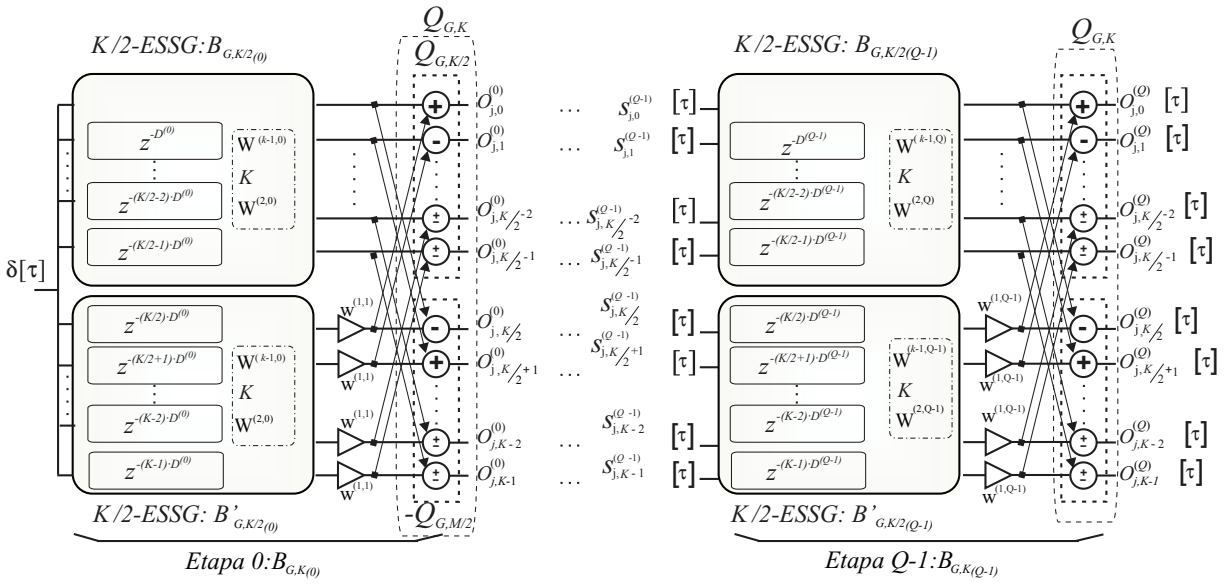


Figura 3.3: Diagrama de bloques del generador eficiente de K-CSS (K-ESSG) propuesto por [MUH+07].

reflejada en el tiempo de esa secuencia, puede verse el sistema anterior como un filtro acoplado a las versiones reflejadas en el tiempo de las secuencias complementarias $s_{j,i}^{(q)}[L-1-\tau]$, donde la longitud de las secuencia es $L = K^Q$. Permutando el orden de los retardos $D^{(q)}$ en cada etapa del esquema de generación se obtienen las secuencias reflejadas, y el filtro resultante es equivalente a un correlador acoplado a las secuencias directas. Este filtro, denominado correlador eficiente de CSS, ESSC (*Efficient Set of Sequences Correlator*), posee K salidas que corresponden a la correlación simultánea de la señal de entrada con las K secuencias complementarias de un conjunto $\mathbf{S}_j^{(k,q+1)}$, como se puede observar en la Figura 3.6 del diagrama en bloques del ESSC.

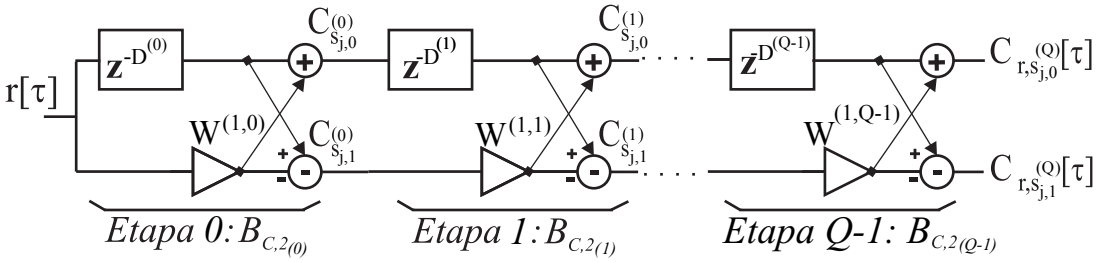


Figura 3.4: Diagrama de bloques del correlador eficiente de 2-CSS (2-ESSC) propuesto por [Bud91, Pop99].

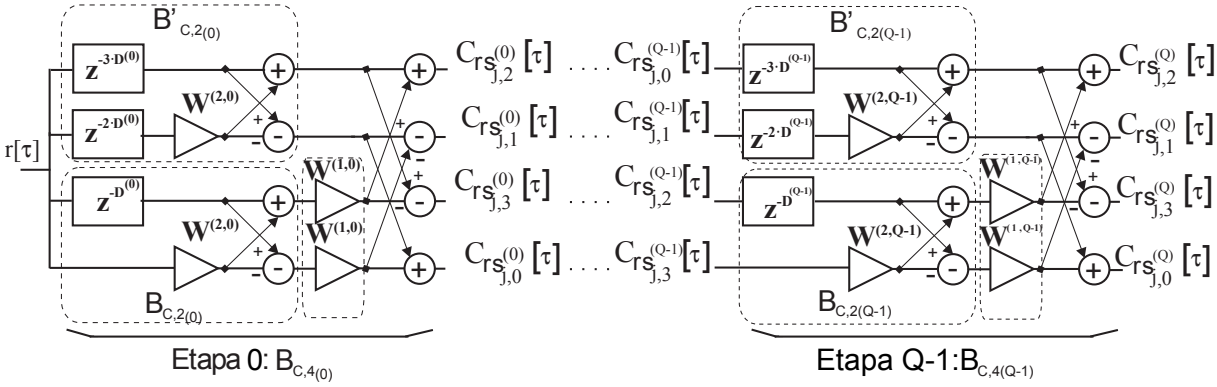


Figura 3.5: Diagrama de bloques del correlador eficiente de 4-CSS (4-ESSC) propuesto por [ÁUM+04].

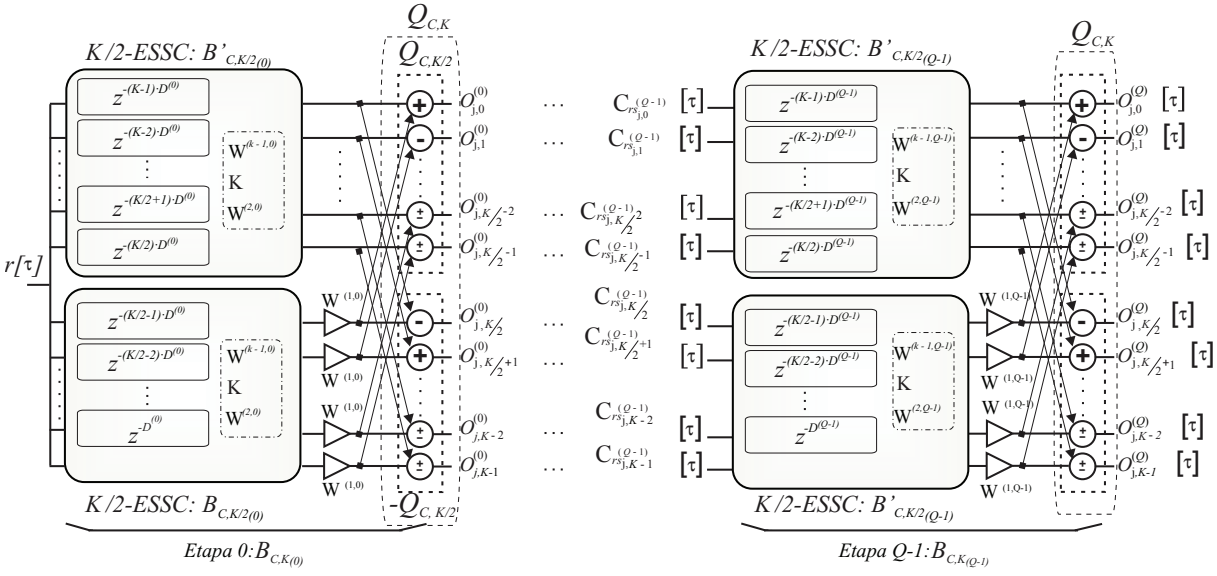


Figura 3.6: Diagrama de bloques del correlador eficiente de K-CSS (K-ESSC) propuesto por [MUH+07].

3.2.2. Correlador eficiente para CSS multinivel

A partir de la arquitectura propuesta por De Marziani *et al.* en [MUH+07], y considerando la adaptación de [PUH+07], se obtiene la arquitectura del generador eficiente para $K_{|MultCSS} = 4$ de CSS multinivel, de longitud $L_{|MultCSS} = K^{Q}_{|MultCSS}$ como se observa en la Figura 3.7. Aquí se pueden apreciar los multiplicadores $A^{(q)}$ en las ramas cruzadas antes de los sumadores, ésta es una de las modificaciones a tener en cuenta para la implementación hardware.

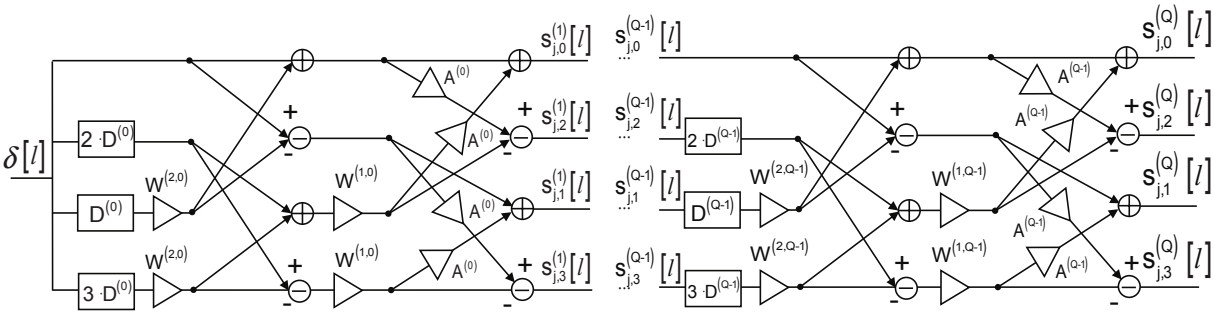


Figura 3.7: Arquitectura del algoritmo de generación de 4-CSS multinivel, $0 \leq j \leq 3$ [GUG13].

Con la arquitectura modular del generador CSS multinivel presentado anteriormente, se pueden generar $K_{|MultCSS}$ conjuntos CSS multinivel para cualquier $K_{|MultCSS} = 2^k, k \in \mathbb{N} - \{0\}$. Consecuentemente, es posible diseñar una arquitectura modular para la correlación de $K_{|MultCSS}$ conjuntos CSS multinivel a partir del generador propuesto, si se modifica para que genere la secuencia inversa. Esto se realiza intercambiando el orden de los retardos en cada etapa del generador de la Figura 3.7, resultando en el correlador de la Figura 3.8, donde los términos $\phi_{r,s^{(q)}_{j,i}}[l]$ son resultados intermedios, diferentes de la función de correlación aperiódica. El resultado solo aparece al final de la etapa.

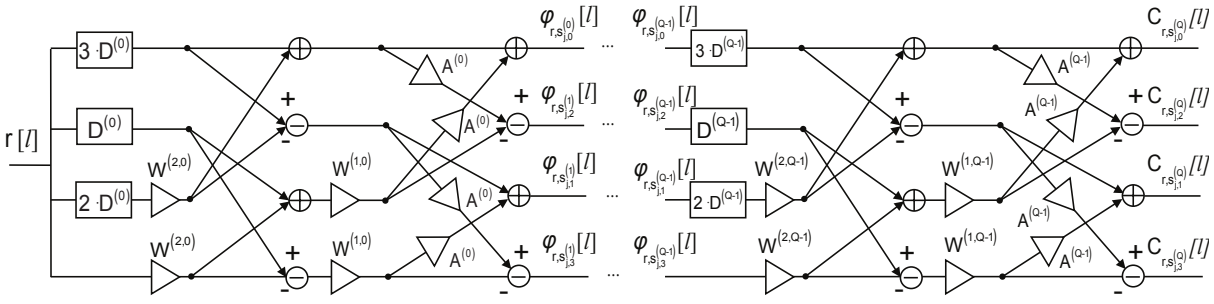


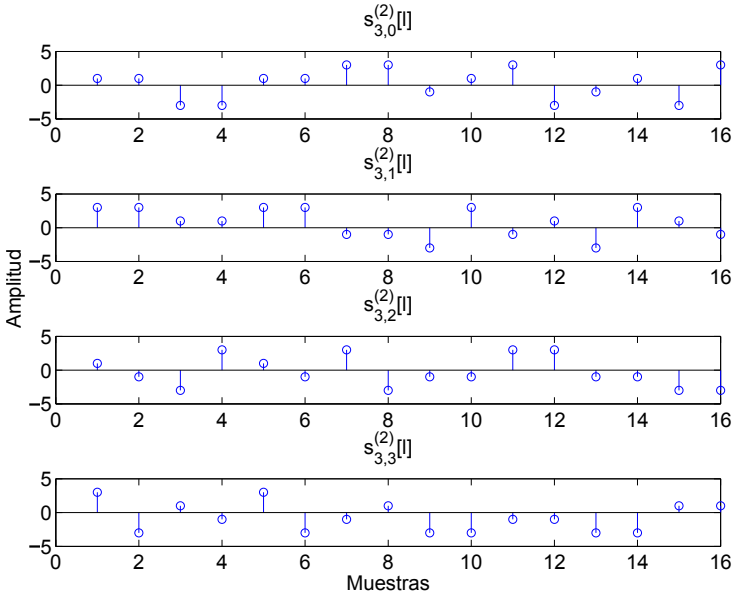
Figura 3.8: Arquitectura del correlador para $K_{|MultCSS} = 2^k$ $(k = 2)$ conjuntos de secuencias complementarias multinivel [GUG13].

Siguiendo con el ejemplo al final de la sección 3.1, si se aplicara al correlador CSS Multinivel, los mismos parámetros con los que se generó las secuencias en la ecuación (3.11) (véase Figura 3.9(a)), se obtendría a la salida del correlador acoplado a estas secuencias multinivel, los resultados mostrados en la Figura 3.9(b).

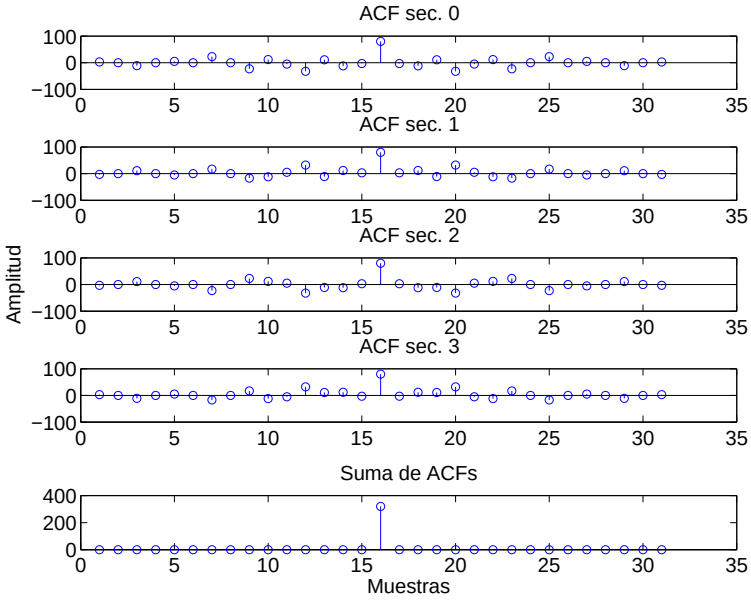
La arquitectura eficiente propuesta por E. García *et al.*, permite reducir la carga computacional necesaria para generar/correlar conjuntos CSS multinivel en comparación con la arquitectura de correlación directa. La eficiencia en la arquitectura presentada se debe gracias a la matriz de retardos del algoritmo de Wornell que reduce el número de etapas necesarias para una longitud dada $L = K^{Q}_{|MultCSS}$. En la Tabla 3.2, se observa una comparación de las operaciones necesarias entre un correlador directo (véase Figura 3.10) y el correlador eficiente CSS multinivel.

	Implementación	Multiplicaciones	Sumas
Cualquier $K_{\|MultCSS}$	Correlador directo	$K^{Q}_{\|MultCSS}$	$K^{Q}_{\|MultCSS} - 1$
$K_{\|MultCSS} = 2^k$	Correlador eficiente	$Q{\cdot}K_{\|MultCSS}$	$Q{\cdot}K_{\|MultCSS} \cdot \log_2(K_{\|MultCSS})$

Tabla 3.2: Operaciones necesarias para la generación y/o correlación, de un CSS multinivel de $K_{|MultCSS}$ secuencias, con la arquitectura directa y el correlador eficiente propuesto.



(a) 4 secuencias de un conjunto complementario multinivel, generada con los parámetros del ejemplo dado.



(b) Funciones de AC de las secuencias generadas y su respectiva suma SACF con propiedades ideales.

Figura 3.9: Secuencias generadas con los parámetros $Q = 2$, $D^{(q)} = \{4^1, 4^0\}$, $A^{(q)} = \{1, 3\}$ y las semillas $W^{(k,q)} = \{[-1, -1], [-1, -1]\}$ del ejemplo al final de la sección 3.1, resultados de AC y SACF.

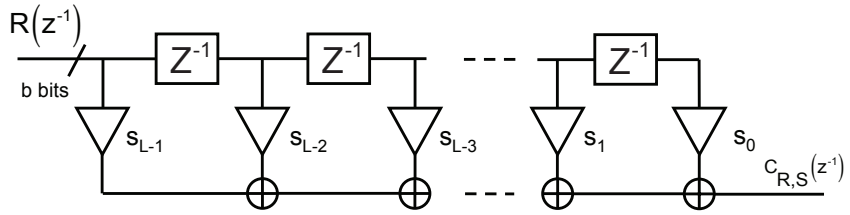


Figura 3.10: Esquema de un correlador directo.

3.3. Pares complementarios ternarios óptimos

A partir de la arquitectura propuesta por E. García *et al.* para generar conjuntos CSS multinivel, es posible generar pares[2] de conjuntos complementarios ternarios óptimos como los encontrados en [GL94, GS01], cuyos valores normalizados pueden ser $\{-1, 0, +1\}$. El término óptimo se refiere a que las secuencias ternarias generadas tienen la mínima cantidad posible de elementos cero. Esto es posible, ya que los retardos $D^{(q)}$ pueden tomar cualquier permutación del conjunto $\{K^0_{|MultCSS}, K^1_{|MultCSS}, K^2_{|MultCSS}, \ldots, K^{Q-1}_{|MultCSS}\}$, o cualquier valor $D^{(q)} > 0$ tal que el CSS multinivel generado tenga una longitud igual a $L_{|MultCSS} = (K_{|MultCSS} - 1) \cdot \sum_{q=0}^{Q-1} D^{(q)} + 1$, los valores de las semillas pueden ser $W^{(1,q)} \in \{-1, +1\}$ y que las amplitudes de los multiplicadores $A^{(q)}$ pueden ser cualquier valor real.

Estas secuencias ternarias, siguen siendo complementarias, se pueden generar de forma sencilla utilizando multiplicadores $A^{(q)} = 1/2$, y permiten generar o correlar secuencias de un mayor número de longitudes.

En la Tabla 3.3 se muestran algunos de los resultados obtenidos de pares ternarios óptimos hasta la longitud 12, a modo de simplificación se asume que los elementos con signos + y − tienen módulo unitario.

Length	Pair	Q	$A^{(q)}$	$W^{(1,q)}$	$D^{(q)}$
3	$++-$ $+0+$	2	$\{\frac{1}{2}, +1\}$	$\{+1, +1\}$	$\{1, 1\}$
5	$+-0++$ $-+0++$	2	$\{-1, -1\}$	$\{+1, +1\}$	$\{1, 3\}$
6	$++-0-+$ $++0+-$	3	$\{+1, -\frac{1}{2}, +1\}$	$\{+1, +1, +1\}$	$\{1, 2, 2\}$
7	$+--0+0+$ $+0+0++-$	3	$\{+1, -\frac{1}{2}, +1\}$	$\{+1, +1, +1\}$	$\{4, 1, 1\}$
9	$+++-0++-+$ $+++-0---+-$	3	$\{+1, +1, +1\}$	$\{+1, +1, +1\}$	$\{1, 2, 5\}$
11	$+---000-+--$ $--+-000+++-$	3	$\{-1, +1, -1\}$	$\{+1, +1, +1\}$	$\{7, 2, 1\}$
12	$+-+++-+00-+++$ $+-++00+++---$	4	$\{+1, +1, -\frac{1}{2}, +1\}$	$\{-1, +1, +1, +1\}$	$\{1, 2, 4, 4\}$

Tabla 3.3: Pares complementarios ternarios óptimos, hasta longitud 12 dados en [GL94, GS01].

Otra de las aplicaciones de estos códigos, es la utilización de los mismos como *kernels*, es decir que se pueden utilizar para generar secuencias de mayor longitud aumentando el número de etapas y modificando los parámetros del generador, para eso definimos los siguientes parámetros:

- Q_{kernel}, es el número de etapas del *kernel* a utilizar.

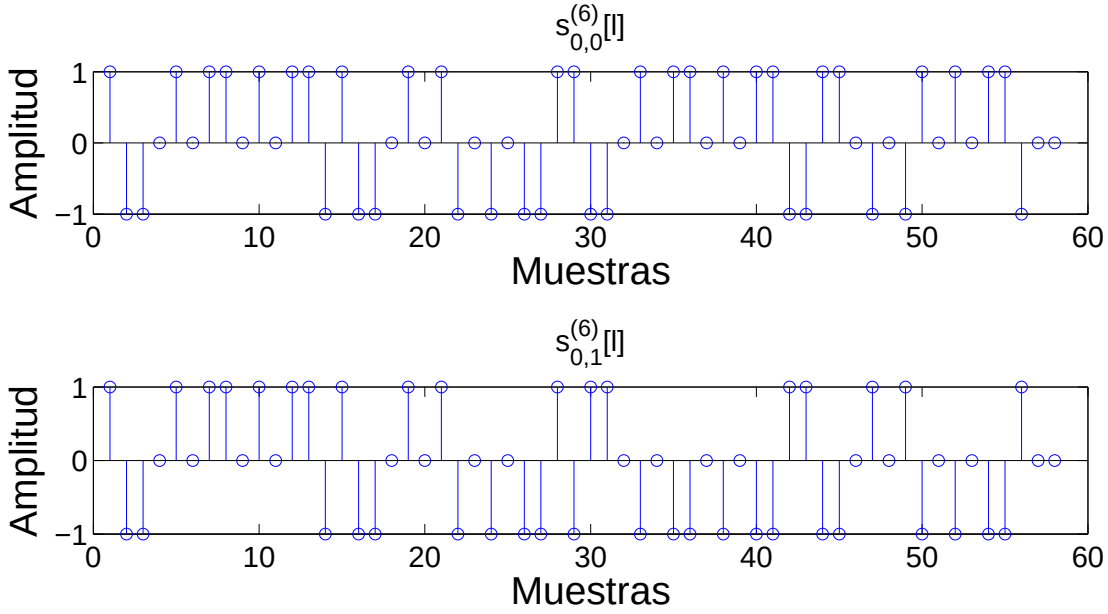
[2] Aquí, para $W^{(k,q)}$ se considera $k = 1$, ya que se generan pares de secuencias ternarias.

- L_{kernel}, es la longitud del par ternario a utilizar como *kernel*.
- $Q = Q_{kernel} + Q_{extra}$ es el número de etapas del nuevo código, siendo Q_{extra} las etapas extra que se añaden aparte de las que tiene el *kernel*.
- $D^{(q)}_{kernel}$ son los retardos del *kernel* a utilizar.
- Los retardos del nuevo código se definen como:
  $D^{(q)} = \left\{ D^{(0)}_{kernel}, \ldots, D^{(Q_{kernel}-1)}_{kernel}, L_{kernel} \cdot 2^0, L_{kernel} \cdot 2^1, \ldots, L_{kernel} \cdot 2^{(Q-Q_{kernel}-1)} \right\}$
- L es la longitud del código nuevo, que se obtiene como $L = \sum_{q=0}^{Q-1} D^{(q)} + 1$. Nótese que en este caso la longitud depende del *kernel* utilizado, y el número de etapas definido. Otra cuestión a tener presente, es que como los retardos ya no se eligen como una permutación del conjunto $\{K^0, K^1, K^2, \ldots, K^{Q-1}\}$, ya no se cumple que $L = K^Q$, y por tanto el número de etapas necesarias para obtener esa longitud aumenta en comparación de utilizar como retardos cualquier permutación del conjunto antes mencionado.
- $A^{(q)}$ y $W^{(1,q)}$ son las amplitudes de cada etapa y los valores de la semilla del nuevo código a generar, éstos se mantienen igual a los que se utilizan para el kernel, y para los elementos de las etapas extra simplemente se les asigna un 1, tanto al valor de la amplitud como a la semilla.

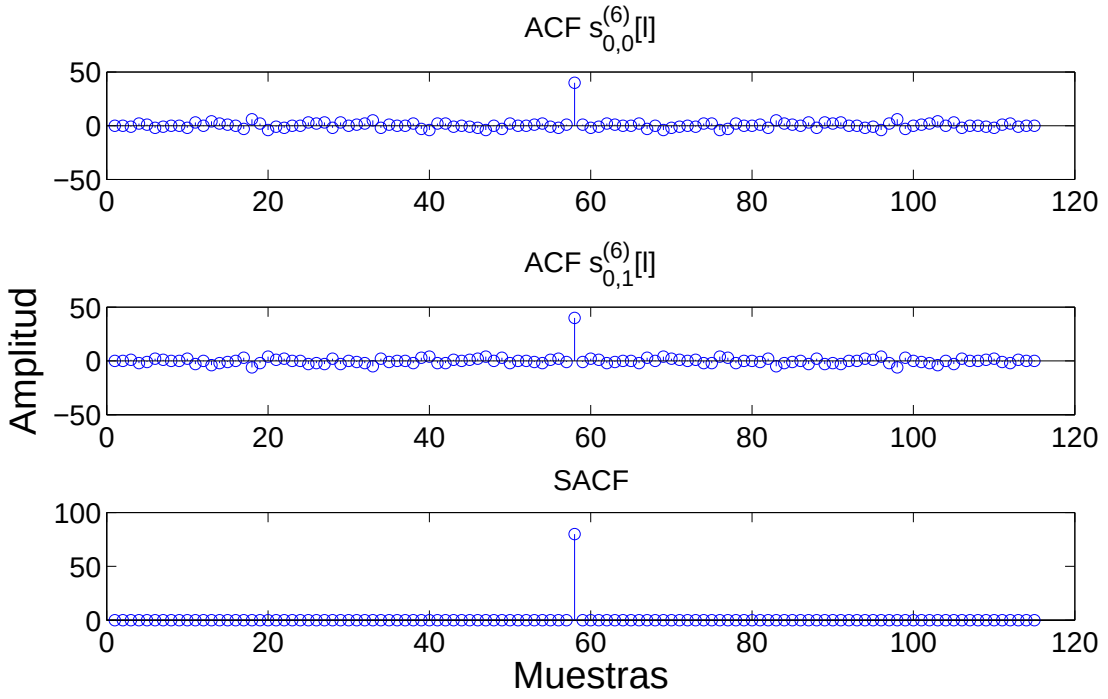
Para entenderlo más fácilmente, a continuación se demuestra con un ejemplo.

Ejemplo. A partir del par ternario de longitud 7, cuyos parámetros, extraídos de la Tabla 3.3, son los siguientes: número de etapas $Q_{kernel} = 3$, amplitud de los multiplicadores en cada etapa $A^{(q)}_{kernel} = \left\{+1, -\frac{1}{2}, +1\right\}$, valores de las semillas en cada etapa $W^{(1,q)}_{kernel} = \{+1, +1, +1\}$ y como retardos $D^{(q)}_{kernel} = \{4, 1, 1\}$. Si se quisiera utilizar este par ternario como *kernel*, y aumentar el número de etapas (por ejemplo a $Q = 6$) para extender la longitud del código, los nuevos parámetros para el generador serían los siguientes: número de etapas $Q = 6$, amplitud de los multiplicadores en cada etapa $A^{(q)} = \left\{+1, -\frac{1}{2}, +1, +1, +1, +1\right\}$, valores de las semillas en cada etapa $W^{(0,q)} = \{+1, +1, +1, +1, +1, +1\}$, y como retardos $D^{(q)} = \left\{4, 1, 1, 7 \cdot 2^0, 7 \cdot 2^1, 7 \cdot 2^2\right\} = \{4, 1, 1, 7, 14, 28\}$. Nótese que tanto para $A^{(q)}$ como para $W^{(1,q)}$ se completan con unos los elementos de las etapas extra. De esta manera, la longitud del código se extiende a $L = 56$ a partir de un código con longitud $L_{kernel} = 7$.

En la Figura 3.11, se pueden observar las secuencias que se generaron a partir de extender aquellas obtenidas con kernel de longitud $L_{kernel} = 7$. Obsérvese que se obtiene un conjunto de secuencias ternarias de longitud $L = 58$, cuya suma de las funciones de autocorrelación cumple con las propiedades ideales. Si se quisiera generar el conjunto complementario incorrelado con este último, habría que utilizar como valores de semilla $W^{(1,q)} = \{-1, +1, +1, +1, +1, +1\}$ donde solo cambia el valor para la etapa $Q = 0$, en las otras 5 se mantiene igual.



(a) Par de secuencias ternarias de longitud $L = 58$, generadas a partir de un *kernel* de par ternario con $L_{kernel} = 7, Q_{kernel} = 3$.



(b) Funciones de AC de las secuencias generadas y su respectiva suma SACF con propiedades ideales.

Figura 3.11: Secuencias ternarias generadas con los parámetros $Q = 6$, $D^{(q)} = \{4, 1, 1, 7, 14, 28\}$, $A^{(q)} = \{1, -\frac{1}{2}, 1, 1, 1, 1\}$ y las semillas $W^{(k,q)} = \{1, 1, 1, 1, 1, 1\}$ del ejemplo de secuencias ternarias, y en (b) sus auto-correlaciones y la suma de ellas.

Capítulo 4

Implementación Hardware del correlador eficiente de CSS multinivel

En la sección 2.4 se mencionaron las distintas plataformas para la implementación hardware, así como también que existen trabajos previos que han utilizado plataformas FPGA para la implementación de correladores eficientes. Considerando esto, y en función del tipo de problema planteado, se decidió que lo más óptimo era implementar el correlador eficiente de secuencias complementarias multinivel sobre una plataforma FPGA. A continuación se verán las consideraciones de diseño y efectos de la cuantificación, para realizar un modelo de simulación a partir del cual definir la mejor manera de cuantificar el dato antes de diseñar la implementación del correlador. Al final de la sección se observan los esquemas de los componentes diseñados, y comparaciones entre el modelo de simulación y la implementación en FPGA.

4.1. Consideraciones de diseño

El diseño del correlador requiere de un dispositivo en el que sea posible trabajar con varios procesos en paralelo, y puedan implementar gran número de multiplicadores y retardos, ya que se necesitan implementar varios de estos componentes según el número de etapas y de secuencias elegido para la arquitectura. Las aplicaciones que se realizan actualmente en el grupo GEINTRA con sistemas LPS (*Local Positioning System*) utilizan 4 balizas, por lo que se necesitarían 4 conjuntos mutuamente incorrelados como mínimo, con longitudes del orden de 1024 bits, considerando esto se realizó un diseño basado en multiplicadores dedicados con los que se pueden obtener estas secuencias fácilmente. De necesitarse conjuntos de secuencias muy largas, o conjuntos con mayor cantidad de secuencias, las multiplicaciones se podrían implementar utilizando descomposición SPT (*Signed Power of Two*) con sumas y desplazamientos [CPG+13]. Como se consideró emplear este tipo de componentes dedicados, es que se decidió utilizar como plataforma una FPGA de gama media/alta de Xilinx, y en función de la disponibilidad de material, se ha optado por una placa de evaluación, *Genesys*, desarrollada por *Digilent Inc.* basada en una FPGA *Xilinx Virtex5 XC5VLX50T* [Xil08].

El diseño desarrollado en VHDL para la plataforma FPGA, es un diseño modular y parametrizable, por lo que tiene en cuenta posibles cambios en el hardware, esto lo hace un diseño genérico y fácilmente adaptable a cualquier otra FPGA con mayor o menor disponibilidad de recursos. Dependiendo de la FPGA elegida serán las restricciones en el número de módulos o etapas del correlador y la longitud de los códigos con los que se trabaje. Por lo que si en un futuro se deseara cambiar de FPGA, se deberían modificar los parámetros del correlador antes

de sintetizar el diseño en VHDL.

En la Tabla 4.1 se muestran los recursos de la familia de FPGAs Virtex 5, en ella se puede observar que el modelo de FPGA a utilizar cuenta con un total de 48 módulos DSP48E, éste componente es el que contiene los multiplicadores dedicados, lo que limitará la cantidad de multiplicaciones a realizar, y por ende el número de etapas a implementar. Este DSP, puede realizar multiplicaciones de 25×18 bits en complemento a dos [Xil12], por lo que introduce otra restricción en cuanto al ancho de bits a la entrada de la sub-etapa que los implementa.

Device	Configurable Logic Blocks (CLBs)			DSP48E Slices(2)	Block RAM Blocks			CMTs(4)	PowerPC Processor Blocks	Endpoint Blocks for PCI Express	Ethernet MACs(5)	Max RocketIO Transceivers(6)		Total I/O Banks(8)	Max User I/O(7)
	Array (Row x Col)	Virtex-5 Slices(1)	Max Distributed RAM (Kb)		18 Kb(3)	36 Kb	Max (Kb)					GTP	GTX		
XC5VLX30	80 x 30	4,800	320	32	64	32	1,152	2	N/A	N/A	N/A	N/A	N/A	13	400
XC5VLX50	120 x 30	7,200	480	48	96	48	1,728	6	N/A	N/A	N/A	N/A	N/A	17	560
XC5VLX85	120 x 54	12,960	840	48	192	96	3,456	6	N/A	N/A	N/A	N/A	N/A	17	560
XC5VLX110	160 x 54	17,280	1,120	64	256	128	4,608	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX155	160 x 76	24,320	1,640	128	384	192	6,912	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX220	160 x 108	34,560	2,280	128	384	192	6,912	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX330	240 x 108	51,840	3,420	192	576	288	10,368	6	N/A	N/A	N/A	N/A	N/A	33	1,200
XC5VLX20T	60 x 26	3,120	210	24	52	26	936	1	N/A	1	2	4	N/A	7	172
XC5VLX30T	80 x 30	4,800	320	32	72	36	1,296	2	N/A	1	4	8	N/A	12	360
XC5VLX50T	120 x 30	7,200	480	48	120	60	2,160	6	N/A	1	4	12	N/A	15	480
XC5VLX85T	120 x 54	12,960	840	48	216	108	3,888	6	N/A	1	4	12	N/A	15	480
XC5VLX110T	160 x 54	17,280	1,120	64	296	148	5,328	6	N/A	1	4	16	N/A	20	680
XC5VLX155T	160 x 76	24,320	1,640	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX220T	160 x 108	34,560	2,280	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX330T	240 x 108	51,840	3,420	192	648	324	11,664	6	N/A	1	4	24	N/A	27	960
XC5VSX35T	80 x 34	5,440	520	192	168	84	3,024	2	N/A	1	4	8	N/A	12	360
XC5VSX50T	120 x 34	8,160	780	288	264	132	4,752	6	N/A	1	4	12	N/A	15	480
XC5VSX95T	160 x 46	14,720	1,520	640	488	244	8,784	6	N/A	1	4	16	N/A	19	640
XC5VSX240T	240 x 78	37,440	4,200	1,056	1,032	516	18,576	6	N/A	1	4	24	N/A	27	960
XC5VTX150T	200 x 58	23,200	1,500	80	456	228	8,208	6	N/A	1	4	N/A	40	20	680
XC5VTX240T	240 x 78	37,440	2,400	96	648	324	11,664	6	N/A	1	4	N/A	48	20	680
XC5VFX30T	80 x 38	5,120	380	64	136	68	2,448	2	1	1	4	N/A	8	12	360
XC5VFX70T	160 x 38	11,200	820	128	296	148	5,328	6	1	3	4	N/A	16	19	640
XC5VFX100T	160 x 56	16,000	1,240	256	456	228	8,208	6	2	3	4	N/A	16	20	680
XC5VFX130T	200 x 56	20,480	1,580	320	596	298	10,728	6	2	3	6	N/A	20	24	840
XC5VFX200T	240 x 68	30,720	2,280	384	912	456	16,416	6	2	4	8	N/A	24	27	960

Tabla 4.1: Tabla de recursos de la familia de FPGAs Virtex-5

A partir del límite de 48 multiplicadores de los que se dispone, y sabiendo que $N_{multiplicadores} = K_{|MultCSS} \cdot Q$, en la Tabla 4.2 se define el número de etapas máximo (Q_{MAX}) que se pueden generar, la cantidad de multiplicadores por etapa (Mult./etapa) y el número de operaciones sumas o restas en las Q etapas (Sumas) dado $K_{|MultCSS}$ conjuntos complementarios.

$K_{\|MultCSS}$	Q_{MAX}	Mult./etapa	Op. Sumas
2	24	2	48
4	12	4	96
8	6	8	144
16	3	16	192
32	1	32	160

Tabla 4.2: Número de etapas máximo, según el limite de multiplicadores disponible para la FPGA XC5VLX50T, dada una determinada cantidad de secuencias $K_{|MultCSS}$. También se muestran las multiplicaciones por cada etapa (Multip.) y las sumas o restas para las Q etapas (Op. Sumas).

Teniendo en cuenta estas consideraciones, se limita el ancho del bus de datos al máximo permitido a la entrada de los multiplicadores, es decir los 25 bits de entrada al multiplicador. Como en cada etapa se realizan sumas y restas, en algún momento estas desbordarán, es por eso que es necesario realizar un estudio de la cuantificación, para detectar los peores casos y saber donde truncar.

4.1.1. Generalidades de la cuantificación

La representación en binario de las secuencias a la entrada del correlador, se define como binario en complemento a dos y de coma fija, ya que los valores de los elementos de las secuencias pueden ser tanto positivos como negativos, y a su vez pueden contener parte decimal. Cuando se habla de representación en coma fija, se representara de la forma $R[e, d]$, donde e es la cantidad de bits asignados a la parte entera, y d a la parte decimal. La suma de ambos es el ancho total del bus de datos. A su vez, de la parte entera, el bit mas significativo, MSB (*Most Significant Bit*), representa el signo del número.

Cuando se trabaja con coma fija, hay que tener en cuenta ciertas cuestiones. Cuando dos números con distinta representación se suman, estos se deben igualar compensando con ceros a la derecha de la parte decimal, o extendiendo el bit de signo a la izquierda de la parte entera. Si tienen la misma cantidad de bits para representar la parte entera, al resultado final se le deberá extender el signo para que no haya desbordamiento, es decir, que el resultado final tendrá un bit más que cada sumando. En las operaciones de multiplicación, el resultado tendrá tantos bits como la suma de los bits de ambos, y se sumaran por separado los correspondientes a la parte entera y a la decimal.

Ejemplo. Para verlo mas claramente, en las expresiones de 4.1 se muestra un ejemplo de cada caso, en (a) se suman dos números con diferente representación, en (b) tienen la misma cantidad de bits para representar la parte entera y en (c) se multiplican dos números con diferente representación.

$$\begin{array}{lll} \quad R[5,2] & \quad R[3,5] & \quad R[5,2] \\ + \quad\quad & + \quad\quad & \times \quad\quad \\ \hline \quad R[3,4] & \quad R[3,4] & \quad R[3,4] \\ \hline \quad R[6,4] & \quad R[4,5] & \quad R[8,6] \\ \text{(a)} & \text{(b)} & \text{(c)} \end{array} \tag{4.1}$$

Cuando se habla de truncar, es en referencia a acortar el número de bits de un determinado número, para pasar de una representación a otra con menos bits. Dado que esto es similar a convertir un número en otro, se debe determinar cual es la mejor manera de hacerlo para distorsionar lo menos posible la información.

4.1.2. Análisis de los efectos de cuantificación a partir de un modelo de simulación

Al momento de realizar el modelo de simulación, se fijaron ciertos parámetros de partida, que pueden ser modificados a conveniencia (dentro de lo permitido por la arquitectura) ya que el modelo es totalmente genérico, y puede ajustarse según las especificaciones de la plataforma final.

- Se define la representación en coma fija de los valores $A^{(q)}$ del multiplicador como $R[3, 2]$, con esta representación los multiplicadores quedan acotados a un rango de valores entre -4 y $+3{,}75$, considerando el MSB como el bit de signo. Estos valores se eligieron ya que son del orden de valores que se emplean en la generación de pares ternarios óptimos.

- Como el multiplicador permite un ancho máximo de datos de 25 bits, se ha ajustado la entrada de la k-ésima sub-etapa a esa cantidad de bits como máximo, definiéndose como $R[25,0]$, con un rango de -2^{24} a $+(2^{24}-1)$. Se considera solo la parte entera, ya que como se trabaja con números relativamente grandes, tanto para valores $A^{(q)}$ de multiplicadores elevados como pequeños, la parte decimal no hace un gran aporte al pico de correlación final. A su vez, esto simplificara el diseño en VHDL.
- La entrada se representa con $R[12,0]$, ya que en la práctica, generalmente los datos se adquieren a través de un Conversor Analógico-Digital (ADC) de 12 bits. El rango de la entrada se acota entre -2048 a $+2047$, aunque para mantener una simetría, realmente se considera entre ± 2047, de esta manera, las secuencias generadas, antes de entrar al correlador, se las normaliza y seguidamente su amplitud es elevada hasta el rango máximo antes mencionado.

En la Figura 4.1 se esquematiza la arquitectura de un correlador 4-CSS Multinivel, con sus componentes principales y las operaciones a realizar. A medida que el número de $K_{|MultCSS}$ conjuntos incrementa, también lo hacen las etapas y sub-etapas. La cantidad de sub-etapas es igual a $k = \log_2(K_{|MultCSS})$, pero solo en la sub-etapa k de cada una de las etapas se tienen los multiplicadores.

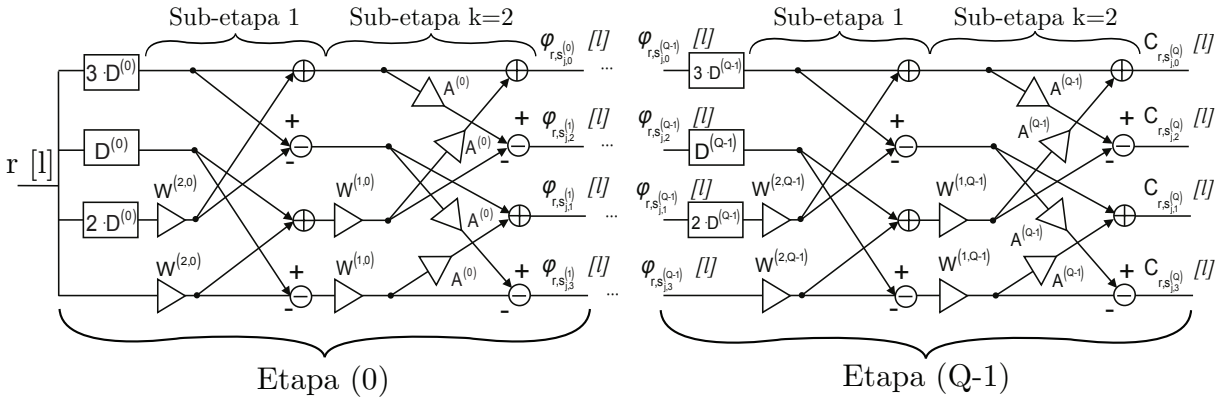


Figura 4.1: Arquitectura del correlador para $K_{|MultCSS} = 2^k$ $(k = 2)$, indicando sus componentes principales.

Esto se debe tener en cuenta para contabilizar el número de bits que irían incrementando según lo explicado en la sección 4.1.1. Entonces, seguido de cada operación suma, se deberá agregar un bit para evitar el desbordamiento, mientras que en las sub-etapas k, se agregarán en total 6 bits, como se describe en la expresión 4.2.

$$\begin{array}{rrl} & R[25,0] & \leftarrow \text{Entrada} \\ \times & R[\ 3,2] & \leftarrow \text{Multiplicador } A^{(q)} \\ \hline & R[28,2] & \\ + & 1,0 & \leftarrow \text{de suma} \\ \hline & R[29,2] & \leftarrow \text{Total} \end{array} \tag{4.2}$$

En la Figura 4.2, se observa un diagrama en bloques de la estructura del correlador y sus componentes básicos. Sobre ella se hace el estudio de como se comportaría el sistema en forma ideal, considerando que: la entrada es de 12 bits, en cada operación de suma se tiene que incrementar un bit y por cada multiplicación 5 bits, para evitar el desbordamiento y la perdida de información. Para analizar un caso puntual se consideró que el número de secuencias $K_{|MultCSS} = 4$, por lo tanto $k = 2$, para $Q = 4$ etapas. Vale mencionar, que los elementos de retardo no incrementan el tamaño del bus de datos.

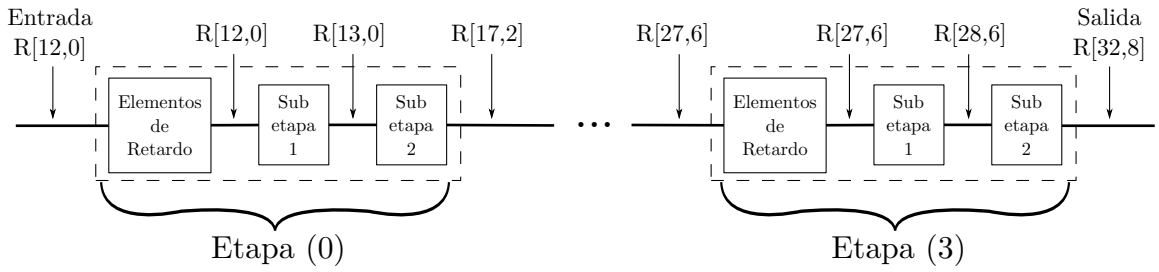


Figura 4.2: Esquema que contempla la cuantificación sin considerar las restricciones. Se consideró un caso puntual para $K_{|MultCSS} = 4$ secuencias, $k = 2$ sub-etapas y $Q = 4$ etapas.

De la gráfica se puede observar, que en el caso planteado idealmente, se necesitaría un total de 32 bits para representar la parte entera y 8 en la parte decimal, esto por cada secuencia $K_{|MultCSS}$ para no perder información por desbordamiento.

Ahora si se consideran las restricciones propuestas, a la entrada del multiplicador se extiende o trunca el dato antes de realizar la multiplicación, y se obtendrían los resultados de la Figura 4.3, en la que se puede observar que la representación máxima descendió a $R[29, 0]$. Los bits de la parte decimal son eliminados al final de la sub-etapa k, ya que estos no suponen una variación considerable en el *Bound* del pico de la SACF, y disminuyen tanto la complejidad del diseño como el consumo de recursos del sistema.

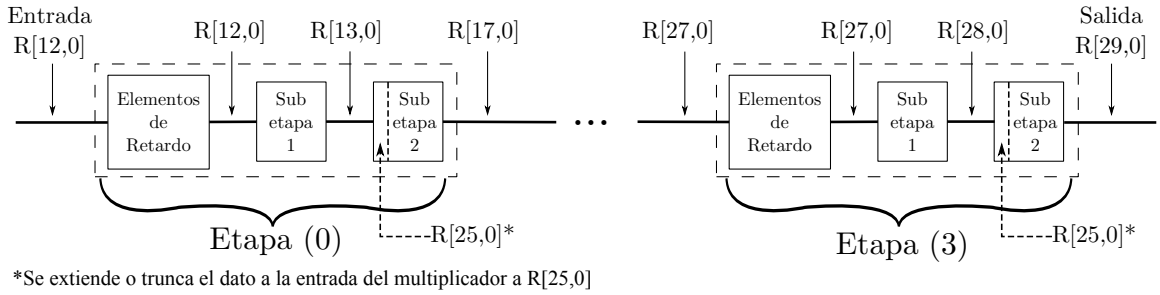


Figura 4.3: Esquema que contempla la cuantificación real, teniendo en cuenta las restricciones propuestas.

Como se vio anteriormente, a la salida de cada sub-etapa se debe aumentar el ancho del bus de datos para no sufrir un deterioro de la información debido al desbordamiento, es por eso que solo se truncan los datos a la entrada de los multiplicadores debido al límite impuesto por ellos, sin embargo en las operaciones de suma, la salida se incrementa en un bit hasta llegar a la k-ésima sub-etapa.

Al momento de definir donde truncar, se plantearon dos casos posibles a analizar:

- El primero donde se consideran mas relevantes los bits de menor peso y se truncan los de mayor peso. (Véase Figura 4.5(a))
- El segundo es la inversa del primero, pero necesita de un cambio adicional para minimizar los errores. A la entrada del sistema, los datos ingresan con 12 bits, pero a continuación, antes de la primer etapa, se extiende el ancho del bus de datos de 12 a 25 bits, esto se realiza reubicando el dato de 12 bits a la izquierda y completando con ceros a la derecha hasta completar los 25 bits. (Véase Figura 4.5(b). Esto es, ya que de lo contrario, en las primeras etapas se perdería más información debido al truncamiento de los bits menos significativos, que es donde se encuentra el dato de entrada. Por lo que ahora el nuevo esquema de cuantificación, siguiendo con el ejemplo anterior, se ve reflejado en la Figura 4.4

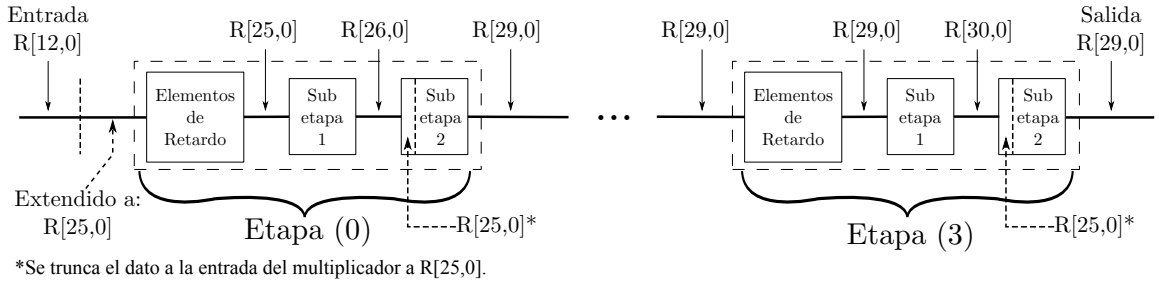


Figura 4.4: Esquema que contempla la cuantificación real, teniendo en cuenta las restricciones propuestas, más la extensión del bus a la entrada.

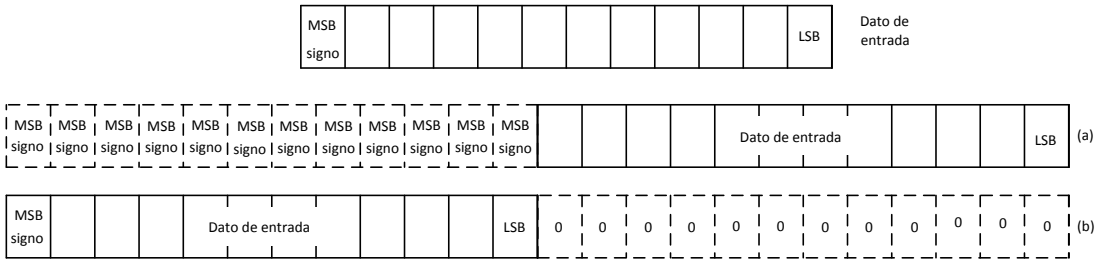


Figura 4.5: Esquema con las dos opciones evaluadas de representación del dato a la entrada, considerando ancho de bus de 25 bits.

A su vez, según la opción de representación del dato a la entrada, se pueden definir algunas opciones de truncamiento, que se describen a continuación:

- Opción 1: En esta opción, se desplaza el dato hacia la izquierda (véase Figura 4.5(b)), y al momento de truncar los datos se eliminan los bits menos significativos.
- Opción 2: Aquí también se desplaza el dato hacia la izquierda, sin embargo, ahora se tienen en cuenta los bits menos significativos más el bit de signo, es decir, 24 bits de datos y uno de signo.
- Opción 3: Para esta opción se mantiene el dato a la derecha (véase Figura 4.5(a)), y se tienen en cuenta los bits más significativos.
- Opción 4: En ésta, también se mantiene el dato a la derecha, pero se tienen en cuenta los bits menos significativos más el bit de signo.
- Opción 5: Aquí se mantiene el dato a la derecha también, pero a diferencia de las anteriores, se tienen en cuenta los bits menos significativos, sin contar con el bit de signo, es decir que cuando la representación ideal supere los 25 bits que limitan la entrada al multiplicador, se perderá la información del signo cuando el dato desborde.

Para hacer más clara la comparación entre opciones, en la Figura 4.6, se puede apreciar de forma visual las 5 opciones evaluadas.

En un caso ideal, el valor esperado del pico de SACF para el desplazamiento nulo, se obtiene según la ecuación 4.3 en el instante $\tau = 0$, donde $s_{j,0}^{(Q)}$ es la secuencia generada a partir de los parámetros Q, $A^{(q)}$ y $W^{(k,q)}$ dados para $K_{|MultCSS}$ secuencias. Sin embargo, en el modelo de simulación esto no ocurre. Como la secuencia generada según estos parámetros, puede tener una amplitud mucho menor, o superar el rango permitido por la representación de los 12 bits a la entrada (véase la Figura 4.7), a estos valores se los normaliza y seguidamente se los ajusta al rango deseado donde pueden valer como máximo ± 2047 (según las restricciones propuestas). No obstante, el correlador tiene cargados los parámetros con los que se generó la secuencia $s_{j,0}^{(Q)}$, con lo que en realidad estaría haciendo la correlación de esta secuencia, con la que tiene niveles de

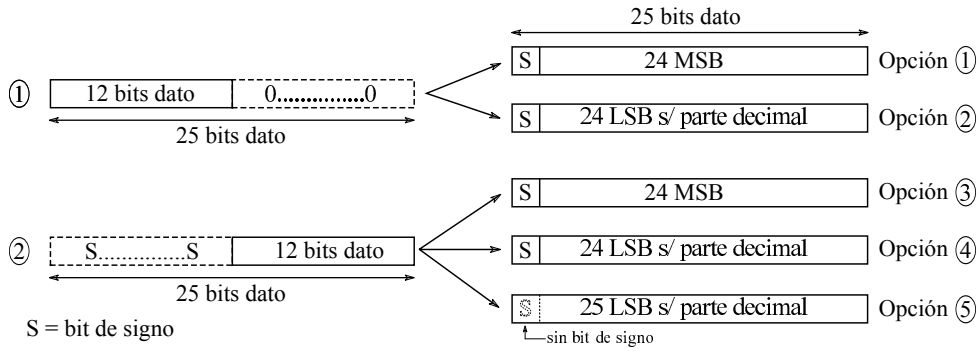


Figura 4.6: Las 5 opciones de truncamiento a evaluar, en forma gráfica.

amplitud ajustados $r[l]$. Por lo que ahora el pico real de correlación esperado, se obtendrá según la expresión 4.4.

$$SACF_{s_{j,0}^{(Q)},s_{j,0}^{(Q)}}[\tau] = \sum_{l=0}^{L-1} \left(s_{j,0}^{(Q)}[l]\right)^2 \cdot K_{|MultCSS} \qquad \tau = 0 \tag{4.3}$$

$$SACF_{r,s_{j,0}^{(Q)}}[\tau] = \sum_{l=0}^{L-1} \left(r[l] \cdot s_{j,0}^{(Q)}[l]\right) \cdot K_{|MultCSS} \qquad \tau = 0 \tag{4.4}$$

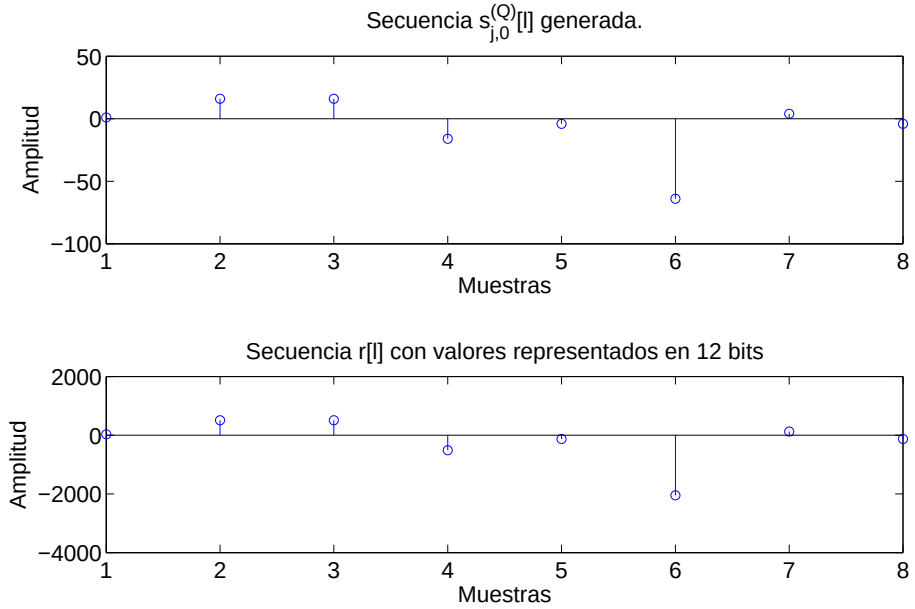


Figura 4.7: Arriba, secuencia generada a partir de parámetros $K_{|MultCSS} = 2$, $Q = 3$, $A^{(q)} = [-4, -4, -4]$. Abajo, misma secuencia pero con valores ampliados al máximo de representación con 12 bits.

Definidas las consideraciones a tener en cuenta, se desarrolló el modelo de simulación sobre el cual se hicieron distintas pruebas para diferentes parámetros de entrada, cambiando el número de secuencias $K_{|MultCSS}$, de etapas Q, diferentes semillas $W^{(k,q)}$, y valores de amplitudes del multiplicador $A^{(q)}$.

A partir de las simulaciones se obtuvieron los resultados tabulados en la sección 6.1. Para verlo más claramente, se muestran gráficas para las 5 opciones de truncamiento, con diferentes

valores de amplitudes, y longitudes de código. Con el fin de analizar el peor caso, se forzaron como valores de los multiplicadores los casos extremos, es decir para $A^{(q)} = -4$, que sería el máximo valor en módulo posible de representar, por tanto para $A^{(q)} = 1/4$, sería el mínimo valor en modulo a representar, y también para un caso intermedio en el que solo uno de los valores es distinto de uno (el segundo valor es $-1/2$), más similar a lo que se utilizaría para la generación de secuencias ternarias. Se menciona como aclaración, que en las gráficas puede haber valores nulos que aparecen en las tablas, pero no son representados en las gráficas debido a la escala logarítmica utilizada.

4.1.3. Resultados obtenidos para la opción 1

Analizando los resultados de la simulación para la primer opción propuesta, se puede observar de la Figura 4.8, que cuando se utilizan valores de multiplicadores pequeños, la suma de correlaciones no es muy buena y comienza a distorsionarse. Otra cuestión que se ve en la Tabla 6.1, es que los picos de la correlación disminuyen su valor a medida que la longitud $L_{|MultCSS}$ del código crece. En la Figura 4.9(b) se puede ver este fenómeno, mientras que en la Figura 4.9(a) se ve como debería quedar la suma de correlaciones ideal.

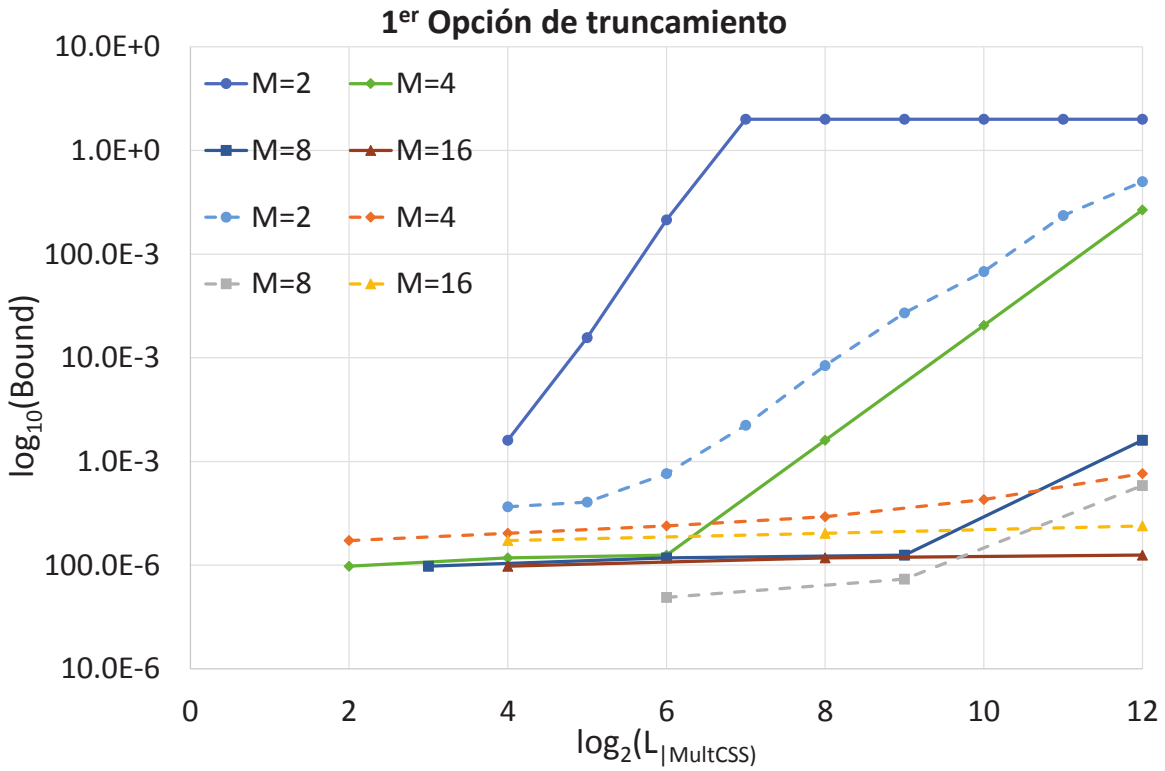


Figura 4.8: Primer opción de truncamiento propuesta. Con linea punteada se representan los valores para multiplicadores $A^{(q)} = -4 \cdot [1\ 1\ \ldots\ 1]$ y con linea sólida se representan los valores $A^{(q)} = 1/4 \cdot [1\ 1\ \ldots\ 1]$.

El efecto de la disminución del pico de SACF a medida que aumenta la longitud de las secuencias, se debe a que inicialmente el dato de entrada es desplazado hacia la izquierda para completar los 25 bits, esto es equivalente a multiplicarlo por $2^{(25-12)}$, es decir se desplaza 13 bits a la izquierda. Posteriormente, en cada operación de suma, el dato obtenido crece en un bit, mientras que en cada multiplicación crece en 3 bits para la parte entera y 2 para la parte decimal, tal como se vio en la expresión 4.2, esto equivale a desplazar el dato hacia la derecha, ya que estos bits se agregan a la izquierda, y del resultado se truncan los bits menos significativos. Por lo tanto, en cada sub-etapa distinta de k se agrega un bit, mientras que en la k-ésima sub-etapa

se agregan 4 bits (los 2 bits que representan la parte decimal de la multiplicación, a la salida no son tenidos en cuenta). Por lo que el valor real del pico de SACF se puede obtener a partir del valor esperado, desplazándolo tantas veces dependiendo de la cantidad de operaciones realizadas por etapa. El número de desplazamientos se define como $Desp = (25-12)-[Q\cdot(3+k)-(k-1)]$, siendo 25 el ancho de bus de datos del multiplicador, 12 el ancho del bus de datos a la entrada del sistema, y 3 bits que se agregan en la k-ésima sub-etapa por la multiplicación. Para verlo más claramente, a continuación se demuestra con un ejemplo

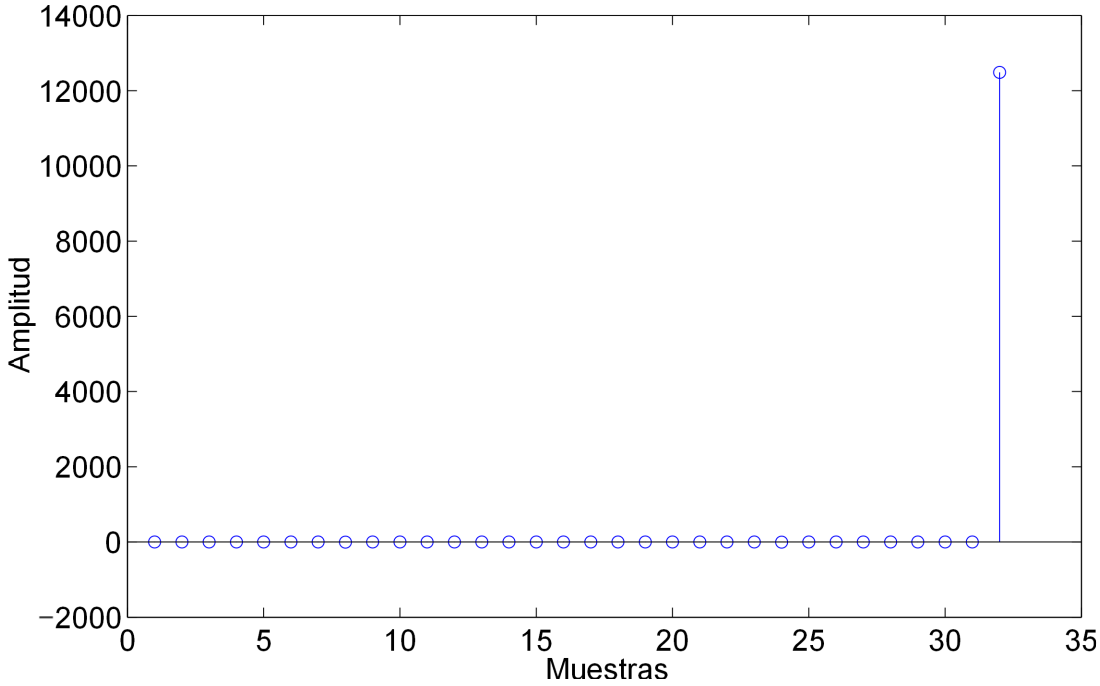
Ejemplo. Si consideramos el caso particular para $K_{|MultCSS} = 4$ secuencias, y $A^{(q)} = -4 \cdot [1, \ldots, 1]$ de las tablas se puede ver que con cada etapa, el valor del pico de SACF disminuye, mientras que el valor esperado va en aumento. A continuación se observa para $Q = 1$, $Q = 2$ y $Q = 3$, el valor del pico de SACF obtenido, y el valor esperado de SACF representados en binario (datos obtenidos de la Tabla 6.1):

$$Q=1 \rightarrow \begin{cases} 0\ 0001\ 0000\ 1111\ 1101\ 1000\ 0000\ 0000 \equiv (17'815'552)_{10} & \text{SACF real} \\ 0\ 000\underbrace{0\ 0000\ 0000}_{\text{-9 bits}}\ 1000\ 0111\ 1110\ 1100 \equiv (\quad 34'796)_{10} & \text{SACF esperado} \end{cases}$$

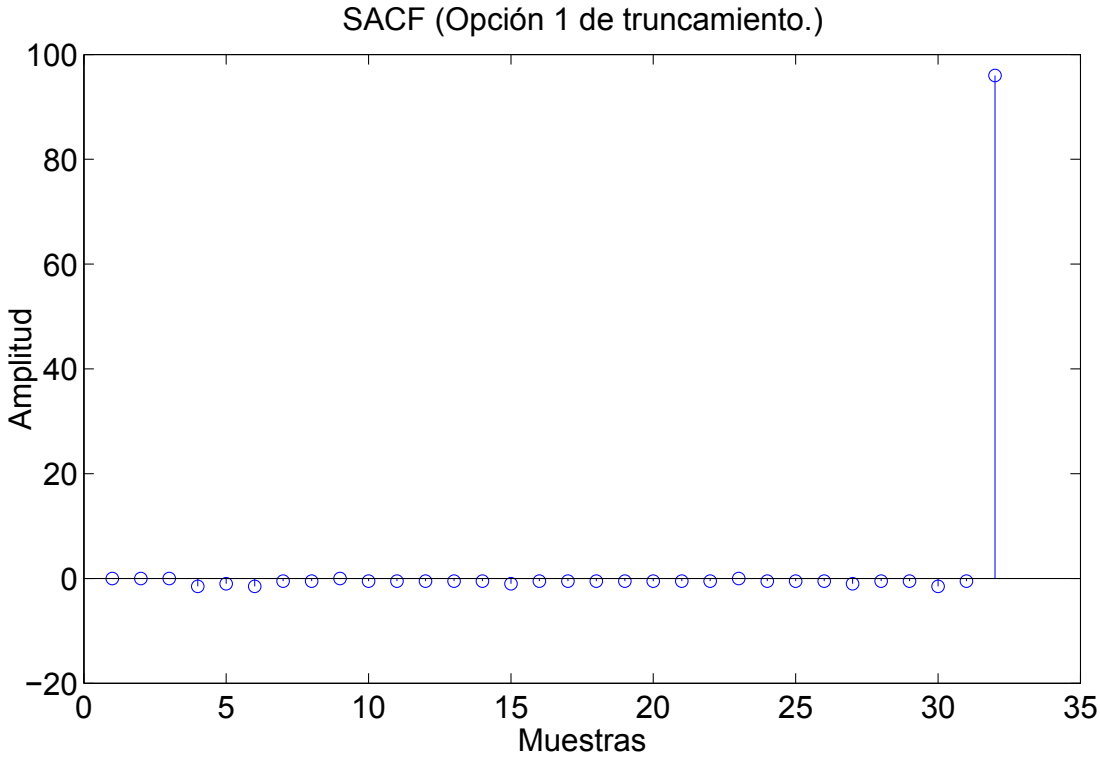
$$Q=2 \rightarrow \begin{cases} 0\ 0000\ 0100\ 1000\ 0011\ 0011\ 1000\ 0000 \equiv (4'731'776)_{10} & \text{SACF real} \\ 0\ 0000\ 0\underbrace{000\ 0}_{\text{-4 bits}}100\ 1000\ 0011\ 0011\ 1000 \equiv (\ 295'736)_{10} & \text{SACF esperado} \end{cases}$$

$$Q=3 \rightarrow \begin{cases} 0\ 0000\ 0001\ 0011\ 0010\ 1101\ 0001\ 1000 \equiv (1'256'728)_{10} & \text{SACF real} \\ 0\ 0000\ 0010\ 0110\ 0101\ 1010\ 0011\ 0000 \equiv (2'513'456)_{10} & \text{SACF esperado} \end{cases}$$

Como se puede observar hay 9 bits de diferencia entre el valor real y el esperado para $Q = 1$, 4 bits de diferencia para $Q = 2$ y 1 bit para $Q = 3$ pero ahora la diferencia es positiva; por lo que entre etapas hay una diferencia constante de 5 bits, eso se debe a las operaciones de suma y multiplicación que se realizan en cada una. Para el caso de $K_{|MultCSS} = 2$ es decir $k = 1$, la diferencia es de 4 bits por cada etapa, 6 bits para $K_{|MultCSS} = 8$ y 7 bits para $K_{|MultCSS} = 16$. Por este motivo, luego de cierta cantidad de etapas, el desplazamiento del valor real hace que comiencen a truncarse más datos relevantes, y su valor disminuye considerablemente.



(a) Suma de las ACF con el correlador sin truncar.



(b) Suma de las ACF con el correlador propuesto para la primer opción de truncamiento.

Figura 4.9: Suma de las ACF, ambas para los parámetros $K_{|MultCSS} = 2$, $Q = 5$ y $A^{(q)} = 1/2 \cdot [1\ 1\ \ldots\ 1]$. Nótese la diferencia de amplitud entre los picos.

4.1.4. Resultados obtenidos para la opción 2

En la Figura 4.10 se ve una de las opciones que presenta muy malos resultados, ya que para longitudes cortas el *Bound* empeora considerablemente respecto a las otras opciones planteadas, y mas cuando se consideran valores pequeños de multiplicador. Esto se debe a que en esta opción se trunca la parte del dato que tiene la información importante, es por eso que se distorsiona rápidamente, y a continuación el pico de correlación es sobrepasado por los lóbulos laterales (véase la Figura 4.11).

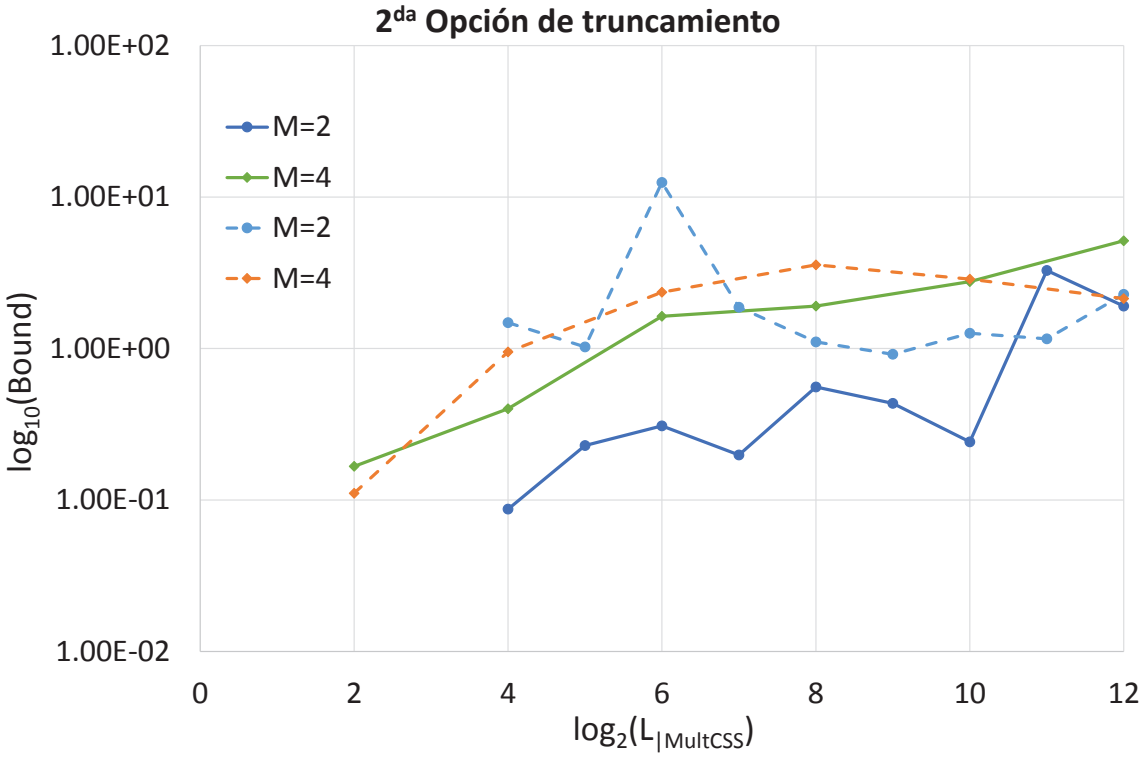


Figura 4.10: Segunda opción de truncamiento. En linea punteada se representan los valores para multiplicadores $A^{(q)} = [-4, -4, -4, -4]$ y con linea sólida se representan los valores $A^{(q)} = 1/4 \cdot [1\ 1\ \ldots\ 1]$.

En la Figura 4.10, solo se representan los resultados para $K_{MultCSS} = 2$ y 4, ya que agregando para $K_{MultCSS} = 8$ y 16 las gráficas se superponen demasiado debido a la gran variación de valores, por lo que no se aprecia claramente el resultado. Estas variaciones, se deben a que la correlación pierde totalmente su sentido por la perdida de información, como se puede observar en la Figura 4.11 para un caso de $K_{|MultCSS} = 4$. En la Figura 4.11 se representa la suma de las ACF para la opción 2. Con estos resultados basta para descartar la posible implementación de esta opción.

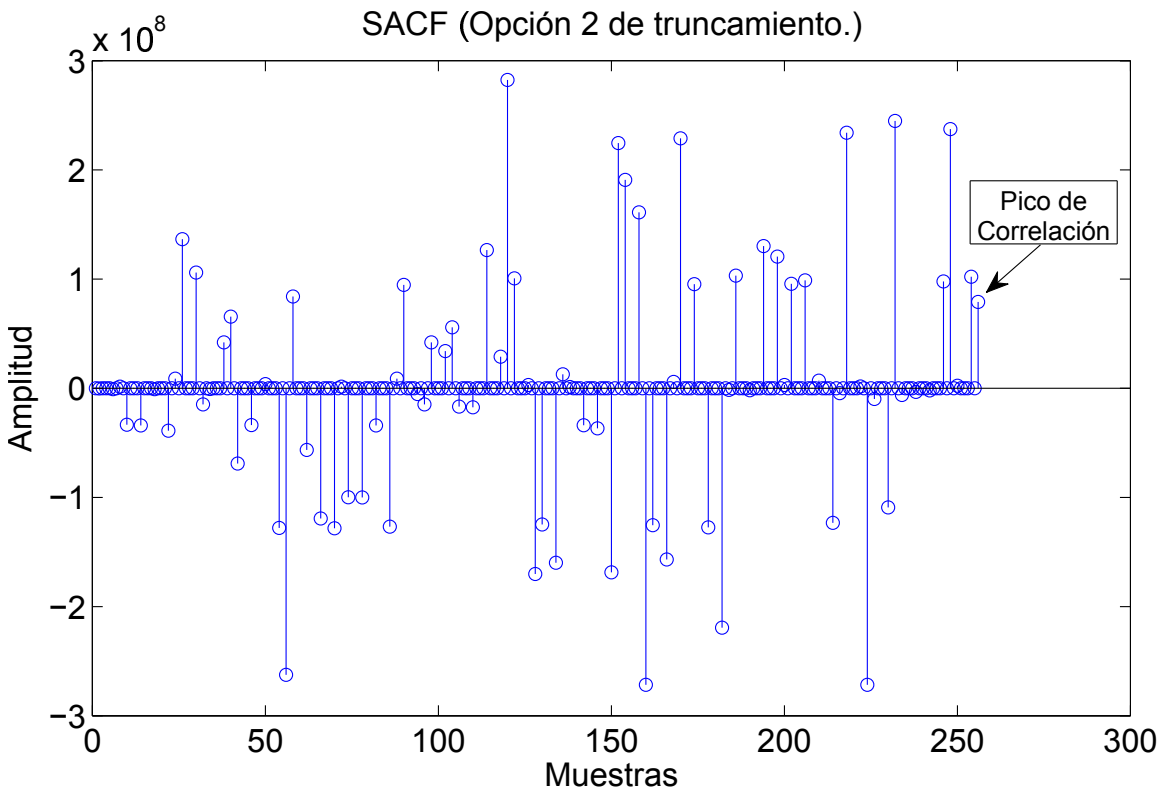


Figura 4.11: Suma de las ACF para la Opción 2 de truncamiento. Se puede observar que el pico de correlación queda enmascarado por los lóbulos laterales. Las secuencias se generaron a partir de $K_{|MultCSS} = 4$, $Q = 4$ y $A^{(q)} = [-4, -4, -4, -4]$.

4.1.5. Resultados obtenidos para la opción 3

En la Figura 4.12 se ve la otra opción que presenta malos resultados, ya que al igual que la anterior, para longitudes menores el *Bound* empeora considerablemente respecto a las otras 3 opciones planteadas. Aquí también se trunca la parte del dato que tiene la información importante. El deterioro de la correlación es más progresivo con cada etapa, hasta que se obtienen lóbulos laterales comparables en amplitud con el pico de correlación (véase Figura 4.13).

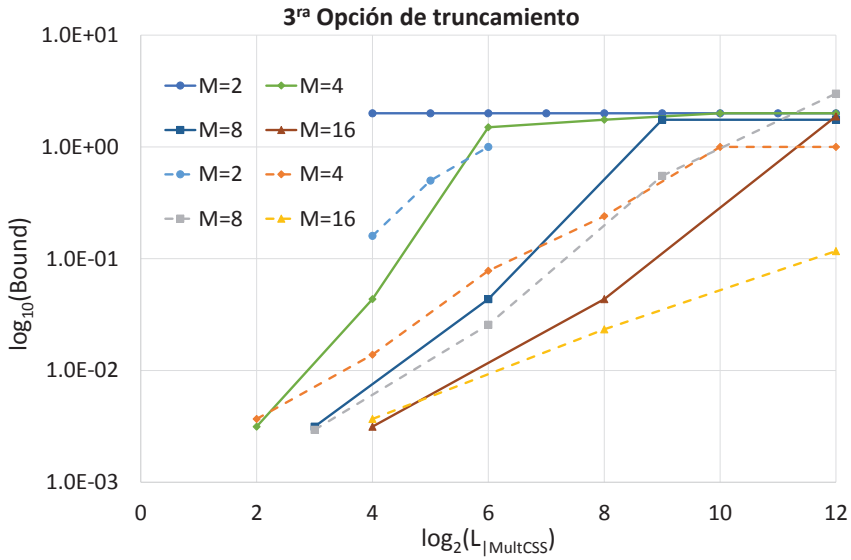


Figura 4.12: Tercera opción de truncamiento. Se representan en linea punteada los valores de *Bound* utilizando multiplicadores $A^{(q)} = [-4, -4, -4, -4]$ y con linea sólida se representan los valores para $A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$.

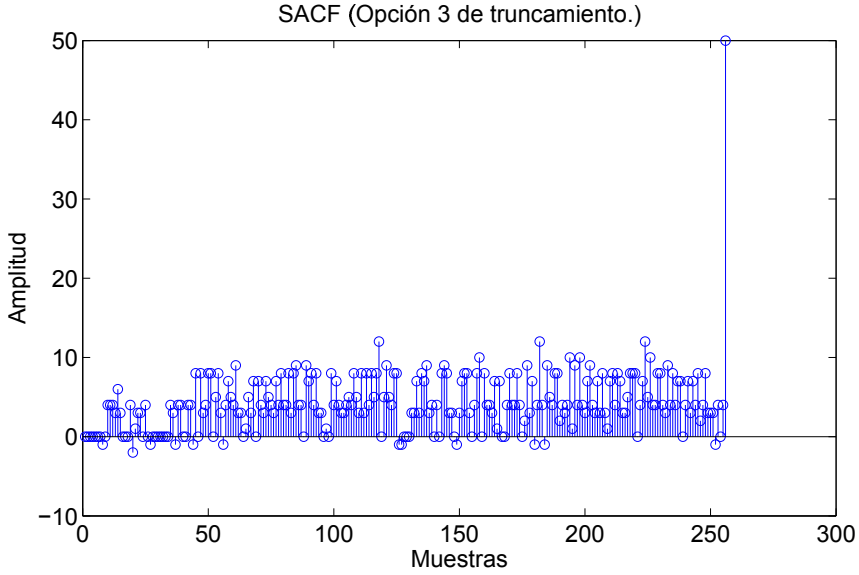


Figura 4.13: Suma de las ACF para la Opción 3 de truncamiento. Aquí la correlación es mejor que en (a), pero sigue teniendo un valor de *Bound* elevado comparado con las otras opciones propuestas. Las secuencias se generaron a partir de $K_{|MultCSS} = 4$, $Q = 4$ y $A^{(q)} = [-4, -4, -4, -4]$.

Para este caso, comparado con la opción 2, muestra para los mismos parámetros de correlación una distorsión menor, sin embargo el *Bound* sigue siendo elevado, por lo que tampoco sería implementable.

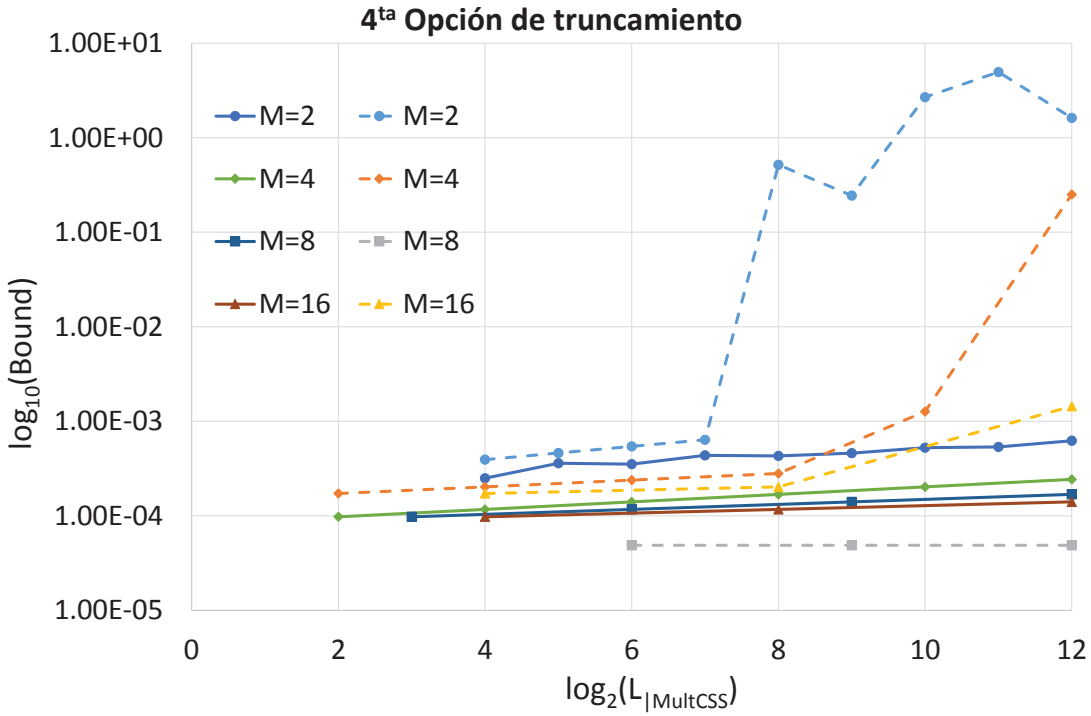
4.1.6. Resultados obtenidos para la opción 4 y 5

Cuando se simulan las opciones 4 y 5 propuestas, ambas presentan muy buenos resultados, con similares valores de *Bound* para longitudes pequeñas, no obstante, para longitudes mayores a 1024 el *Bound* de la opción 4 es apenas más elevado que la otra, como se puede apreciar en las Figuras 4.14(a) y 4.14(b).

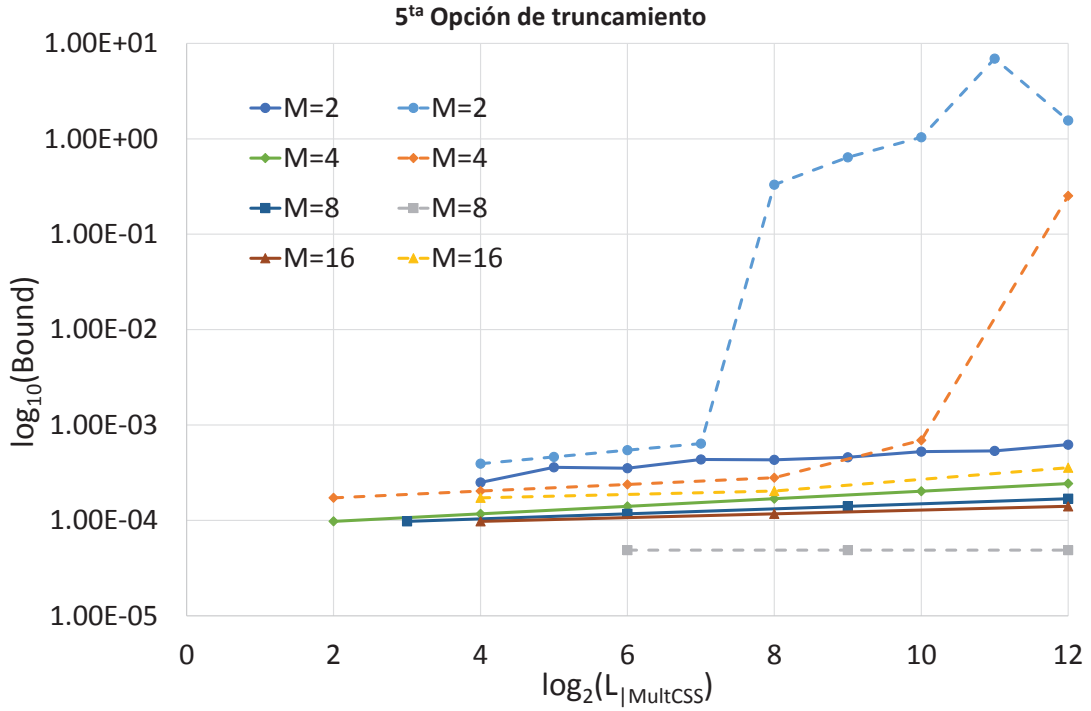
Estos dos casos presentan muy buenos valores de correlación, e incluso iguales para las primeras longitudes, pero ya cuando empiezan a desbordar en longitudes más elevadas, la quinta opción de la Figura 4.14(b) muestra una pequeña mejora frente a la Figura 4.14(a). Los saltos de los valores de *Bound* que se pueden observar, se deben a las longitudes donde el número de bits para la representación supera los 25 bits que limitan la entrada de los multiplicadores, y esto produce una pequeña distorsión en la correlación que aumenta, según lo haga el número de etapas o la longitud de las secuencias.

Empleando la opción 5, cuyos resultados se muestran en la Tabla 6.5, se observan los mejores valores de *Bound*, principalmente para valores de amplitudes pequeñas, que tienen mayor utilidad que los valores grandes de multiplicadores $A^{(q)}$, ya que los valores pequeños son los que se utilizarían para la generación de pares complementarios ternarios óptimos. Esta opción no considera el bit de signo, por lo que utiliza un bit más para la representación del dato, a costa de perder el signo del mismo. Esto produce que cuando la representación de bits ideal supera la representación real limitada por la entrada de los multiplicadores, algunas etapas o iteraciones, sufren una inversión del pico de correlación, con lo que la suma de ellos (SACF) resultará también en un pico negativo, tal como se puede ver en la Tabla 6.5, y en la Figura 4.15(b), esto ocurre tanto para valores positivos como negativos de multiplicadores. A la hora de la implementación este efecto no supondría ningún problema, ya que es posible detectar la SACF con un detector que realice el valor absoluto de la misma.

Tanto de las gráficas como de las tablas, se puede concluir que para cualquiera de las opciones de truncamiento propuestas, se observa un aumento en el valor del *Bound* a medida que la longitud del código crece, o lo que es lo mismo, aumente el número de etapas.

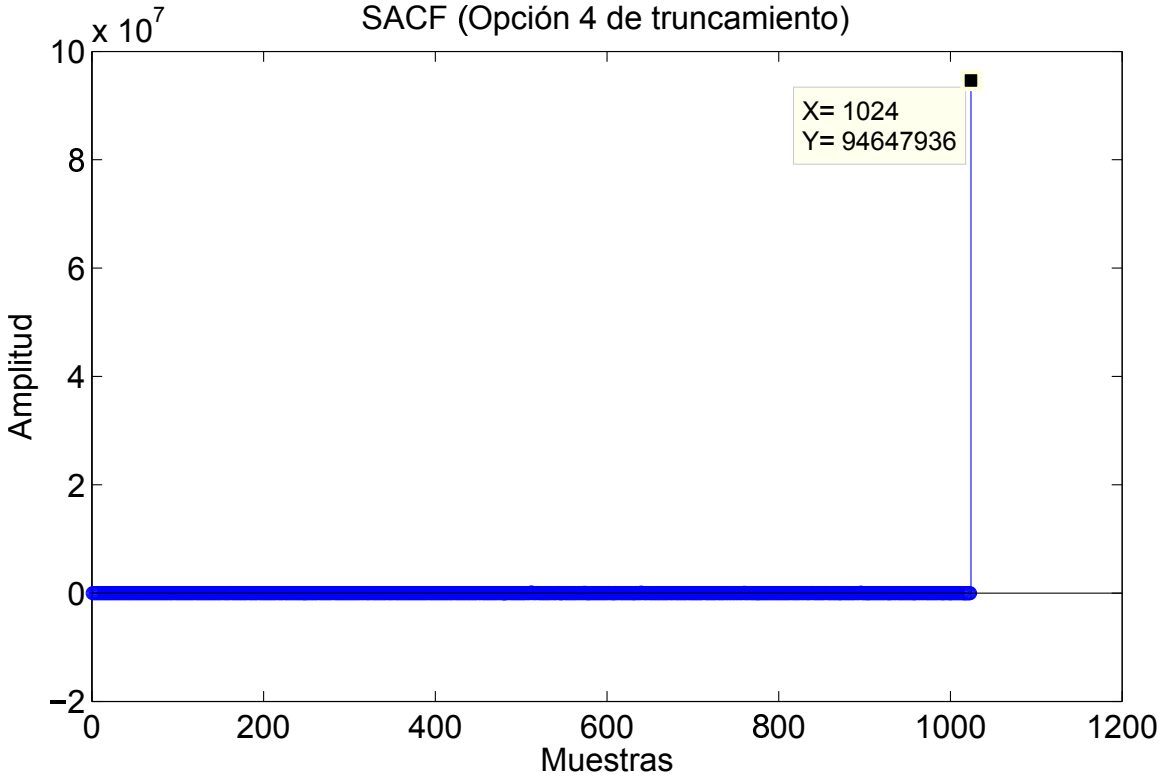


(a) Cuarta opción de truncamiento. Se representan en linea punteada los valores de *Bound* utilizando multiplicadores $A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$ y con linea sólida se representan los valores para $A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$.

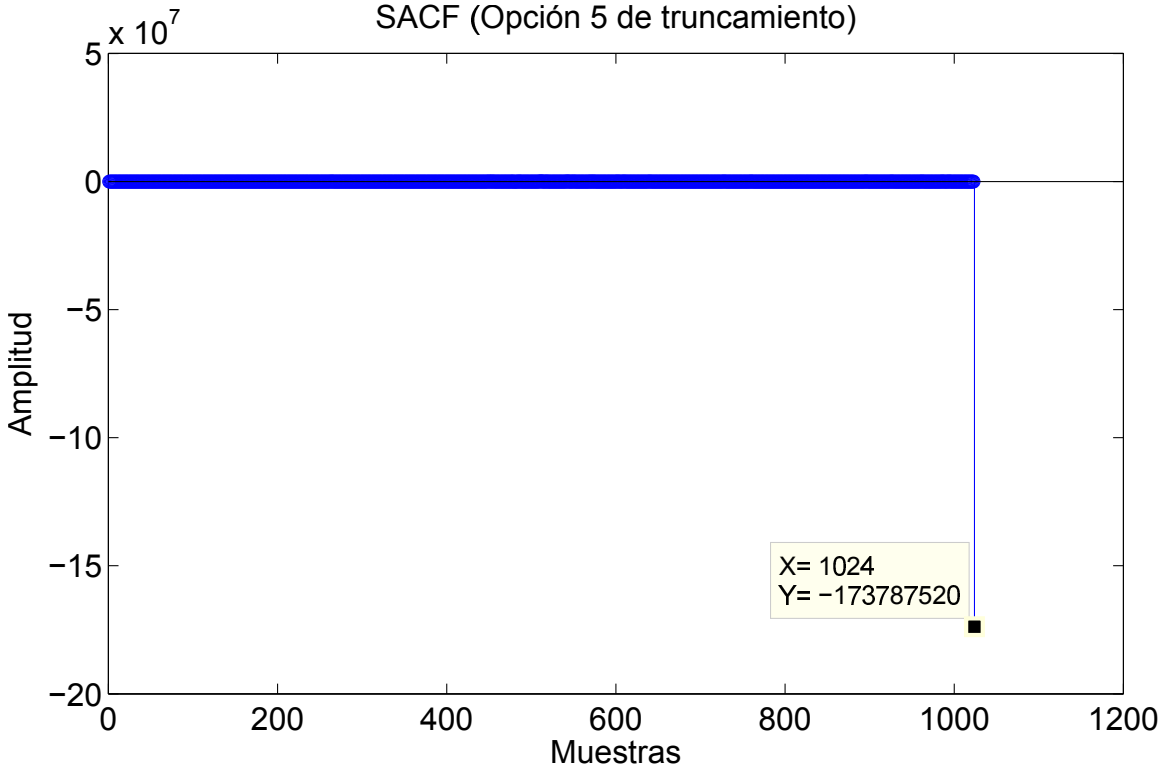


(b) Quinta opción de truncamiento. En linea punteada se representan los valores para multiplicadores $A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$ y con linea sólida se representan los valores para $A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$.

Figura 4.14: Representación del *Bound* frente para diferentes longitudes, obtenidas para las opciones 4 y 5 de truncamiento. En ambas gráficas hay valores nulos que no son representados debido a la escala logarítmica empleada.



(a) Suma de las ACF para la Opción 4 de truncamiento.



(b) Suma de las ACF para la Opción 5 de truncamiento. Como para esta opción no se considera el signo, en las iteraciones impares, aparece el pico de correlación negativo.

Figura 4.15: Suma de correlaciones para los parámetros $K_{|MultCSS} = 4$, $Q = 5$ y $A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$.

4.1.7. Discusión

Cuando se truncan los bits menos significativos y se utilizan los de mayor peso, el pico de correlación va disminuyendo conforme aumenta la longitud de las secuencias, aunque el *Bound* sea menor que para la otra opción de truncamiento, es decir en los casos en que se tienen valores de multiplicadores altos. Esta disminución progresiva del pico de correlación hace que a partir de un cierto número de etapas no sea valida la correlación, debido a su insignificante valor. Sin embargo, truncar los bits más significativos, tiene sentido cuando los valores de los multiplicadores son elevados, ya que permite alcanzar un mayor número de etapas con niveles menores de *Bound*, a costa de tener picos de correlación menores al valor esperado.

Habiendo analizado las distintas posibilidades de representación y opciones de truncamiento del dato antes de los multiplicadores, se consideró que la opción 5 es la más optima para implementar, ya que posee los valores mínimos de *Bound* en todo el rango de representación de los valores de multiplicadores. De todas maneras, estos valores con los que se realizó el estudio son extremos, ya que la secuencia que se obtiene al utilizar estos multiplicadores no tienen aplicación práctica. De hecho, en la práctica solo un par de multiplicadores serán distintos de uno. En la próxima sección se describirá la implementación en la plataforma FPGA del correlador eficiente CSS multinivel, teniendo en cuenta las restricciones aquí estudiadas.

4.2. Detalles de la Implementación Hardware

Una vez simuladas las distintas opciones de truncamiento, y elegida la que se implementará, se procedió al diseño del correlador genérico utilizando dicho método de truncamiento. En la Figura 4.16 se puede observar el esquema del correlador para el caso de $K_{|MultCSS} = 4$. Para el diseño se utilizó un lenguaje de descripción hardware VHDL.

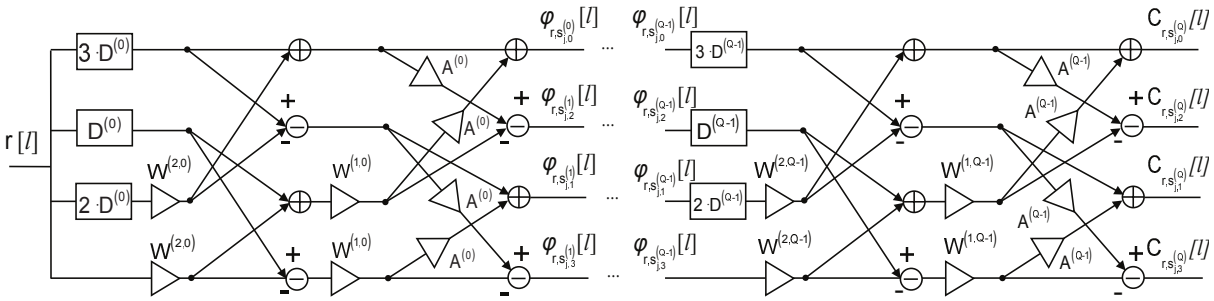


Figura 4.16: Esquema del correlador multinivel a implementar, para el caso $K_{|MultCSS} = 4$.

El presente trabajo se basó en la implementación realizada por M. C. Perez *et al.* del correlador K-CSS binario en [PUH+06], para realizar un diseño que lo generalice llevándolo al alfabeto multinivel. En la Figura 4.17 se observa un esquema del *Top System*, el cual es el bloque/nivel que contiene los componentes necesarios para realizar la correlación. En él se ven los puertos de conexión al exterior de la placa, a partir de ellos se realizará la configuración necesaria para correlar la secuencia previamente generada con los mismos parámetros, y obtener en el puerto de salida la correlación de la secuencia de entrada con las $K_{|MultCSS} = 2^k$ secuencias del conjunto.

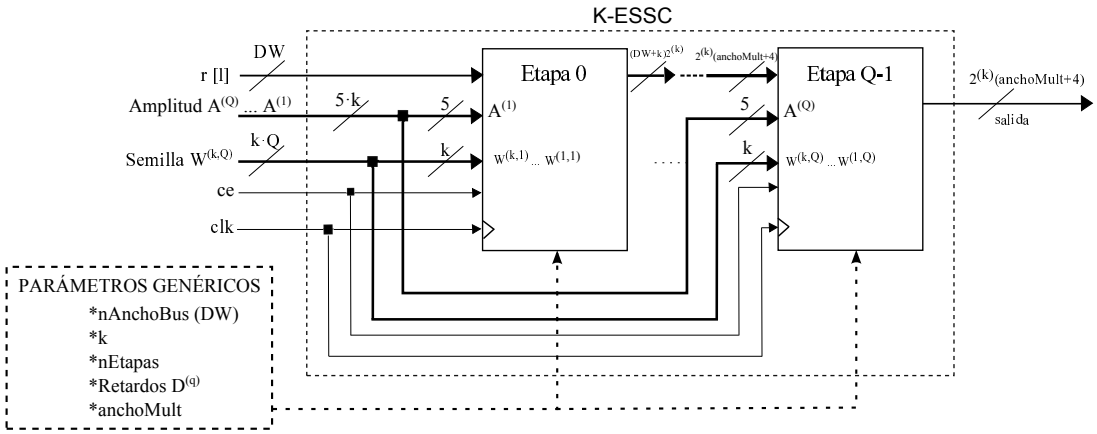


Figura 4.17: Esquema del *Top System* donde aparecen los puertos de entrada y salida al exterior, y parámetros genéricos de configuración.

En este esquema se pueden observar parámetros genéricos a utilizar en el sistema, los cuales deben ser configurados para cada correlación en particular ya que condicionan el diseño a la hora de sintetizar el sistema e implementarlo definitivamente. Si se quisiera que fuera totalmente parametrizable en tiempo de ejecución, los recursos utilizados aumentarían considerablemente, ya que habría que tener reservados un número máximo de elementos que pueden o no utilizarse. A continuación se detallan los parámetros genéricos:

nAnchoBus Este parámetro representa el ancho de bus de entrada en bits del correlador. En este trabajo se utilizarán 12 bits, ya que se supone que la entrada digital viene dada por un conversor ADC de dicha resolución.

k Con este parámetro, se obtiene el número de secuencias del conjunto como $K_{|MultCSS} = 2^k$. A su vez es equivalente al número de sub-etapas que posee cada etapa (véase la Figura 4.18).

Retardos ($D^{(q)}$) Permite ingresar los valores de retardo $D^{(q)}$ de cada etapa. El número de registros de desplazamiento a utilizar, depende de sus valores y del tamaño del bus de datos.

nEtapas Indica el número de etapas del sistema, es equivalente a Q. Se puede definir la longitud de las secuencias a correlar en función de $D^{(q)}$, Q y $K_{|MultCSS}$ como $L_{|MultCSS} = (K_{|MultCSS} - 1) \cdot \sum_{q=0}^{Q-1} D^{(q)} + 1$.

anchoMult Indica el ancho de bits a la entrada del multiplicador, este parámetro permite adaptar el ancho del bus de datos al tamaño máximo del multiplicador dedicado DSP. Este es el condicionante por el que se trunca la entrada de la k-ésima sub-etapa.

En la Figura 4.18 se detalla el esquema de una etapa $(q-1)$, formados por los siguientes bloques: aparece un primer bloque secuencial de retardos que recibe como señales de entrada la correlación parcial de la etapa $(q-2)$ y las del retardo $D^{(q-1)}$ correspondiente; después se incluyen unas etapas de ordenación de entrada y salida; y por último el bloque combinacional que contiene las sub-etapas que realizan las operaciones de suma y resta, y las multiplicaciones para el caso de la k-ésima sub-etapa.

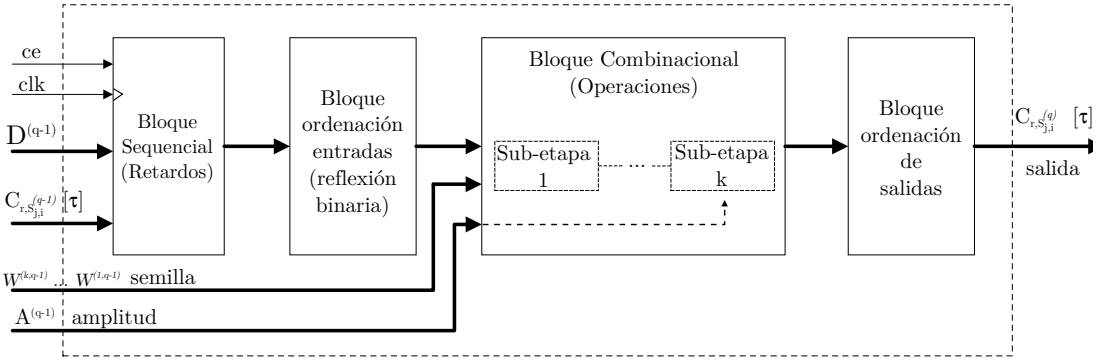


Figura 4.18: Esquema de la etapa $q-1$.

- El primer bloque de la Figura 4.18, comenzando de izquierda a derecha, implementa los retardos $D^{(q)}$ de cada etapa $(0 < q < Q-1)$, según entran las secuencias en el orden $\{(K-1) \cdot D^{(q)}\ ,\ (K-2) \cdot D^{(q)}\ ,\ \ldots\ ,\ 0\}$, donde la última secuencia no es retardada. Esta implementación de los retardos se lleva a cabo a partir de la utilización de registros de desplazamiento serie SRL16 que contienen los *slices*. Cada uno de ellos permite retardar un bit de dato, un máximo de 16 ciclos de reloj, por lo que si el retardo es mayor a 16, se concatenan más registros en serie hasta alcanzar el valor deseado. El valor del retardo en cada etapa, viene dado por el parámetro genérico que se ingresa en el *Top System*.
- A continuación sigue un bloque que reordena las salidas retardadas según el algoritmo de reflexión binaria propuesto en [Pér09], el cual permite obtener una estructura más ordenada en términos de diseño, y facilita la implementación modular de los componentes, sin la necesidad de recursos adicionales. En este algoritmo se considera que los coeficientes que multiplican los retardos (véase la Figura 4.16), se encuentran ordenados de la forma $\{K-1, K-2, \ldots, 1, 0\}$. Si a estos valores se los representa en formato binario con k bits, los retardos se reordenan leyendo su representación binaria al revés. En la Tabla 4.3 se observan como quedarían las salidas para el caso específico de 8-CSS multinivel.
- El bloque combinacional realiza las operaciones necesarias para la correlación en cada etapa, dentro de las cuales, las primeras $k-1$ sub-etapas son solo sumas y restas (véase la

$b_2b_1b_0 \rightarrow$	Reflexión Binaria	$\rightarrow b_0b_1b_2$
(000)	$\rightarrow$	(000)
(001)	$\rightarrow$	(100)
(010)	$\rightarrow$	(010)
(011)	$\rightarrow$	(110)
(100)	$\rightarrow$	(001)
(101)	$\rightarrow$	(101)
(110)	$\rightarrow$	(011)
(111)	$\rightarrow$	(000)

Tabla 4.3: Algoritmo de reflexión binaria para $K_{|MultCSS} = 8$.

Figura 4.18). La configuración final de los sumadores y restadores, dependerá específicamente del valor de las semillas $W^{(k,Q)}$ que pueden tomar valores $[+1, -1]$, como se puede apreciar en la Figura 4.19(a). Para la k-ésima sub-etapa se realizará la multiplicación de la entrada con el valor $A^{(q)}$ correspondiente. Por último, al igual que con las sub-etapas anteriores, el valor de las semillas determinarán la disposición final de los sumadores y restadores (véase la Figura 4.19(b)).

- El último bloque se encarga de reordenar las secuencias de salida, para que mantengan la misma ubicación con la que ingresaron al bloque de retardos. Esto también se realizó a partir de lo propuesto en [Pér09]. Habiendo reordenado las secuencias a la entrada de la etapa con el algoritmo de reflexión binaria, las correlaciones a la salida quedan ordenadas según la siguiente expresión:

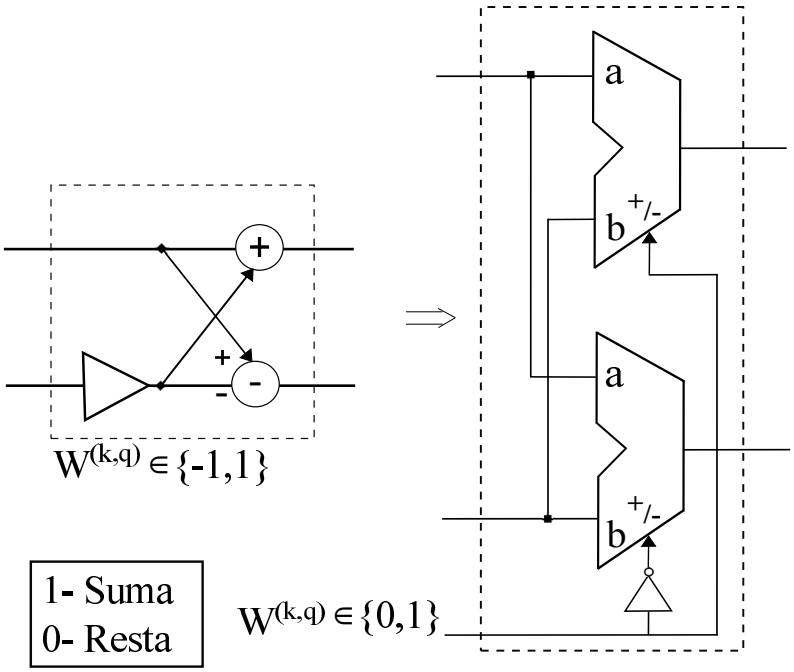
$$\begin{array}{ll} \text{Si } x \text{ es par:} & n = (K_{|MultCSS} - 2 - x) \\ \text{Si } x \text{ es impar:} & n = (K_{|MultCSS} - x) \end{array} \tag{4.5}$$

siendo x la posición de una salida cualquiera en una etapa, considerando $x = 0$ la salida de la rama superior, y n el índice de la correlación parcial $C_{r,s_{j,n}^{(q)}}[l]$ con la secuencia n-ésima. En la Tabla 4.4, se muestran las salidas de la etapa para el caso específico de 4 y 8-CSS multinivel y el índice de correlación correspondiente a esa salida. Por lo que este bloque de ordenación de salidas, se encarga de ubicar las correlaciones en orden ascendente como muestra la siguiente expresión:

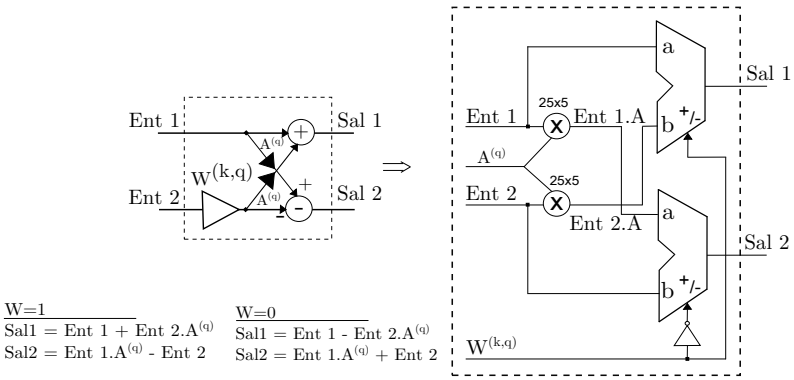
$$\left[C_{r,s_{j,0}^{(Q)}}[l] \; C_{r,s_{j,1}^{(Q)}}[l] \; \ldots \; C_{r,s_{j,K-1}^{(Q)}}[l] \right]^T \tag{4.6}$$

Nº de salida de la etapa x	Índice de la secuencia $s_{j,n}^{(q)}$ complementaria correspondiente	
	$K = 4$	$K = 8$
0	2	6
1	3	7
2	0	4
3	1	5
4	-	2
5	-	3
6	-	0
7	-	1

Tabla 4.4: Correspondencia entre las salidas x de una etapa y la correlación parcial de $C_{r,s_{j,n}^{(q)}}[l]$ obtenida tras aplicar el algoritmo de reflexión binaria, para $K_{|MultCSS}$ 4 y 8 secuencias.



(a) Esquema de implementación de una sub-etapa inicial, que realiza las operaciones de suma y resta en función de las semillas $W^{(k,Q)}$.



(b) Esquema de implementación de la k-ésima sub-etapa, que realiza la operación de multiplicación de la etapa, y seguido de la suma y resta en función del valor de la semilla $W^{(k,Q)}$.

Figura 4.19: Esquemas de implementación en hardware de componentes internos de cada etapa.

4.2.1. Comparativa con el modelo de simulación

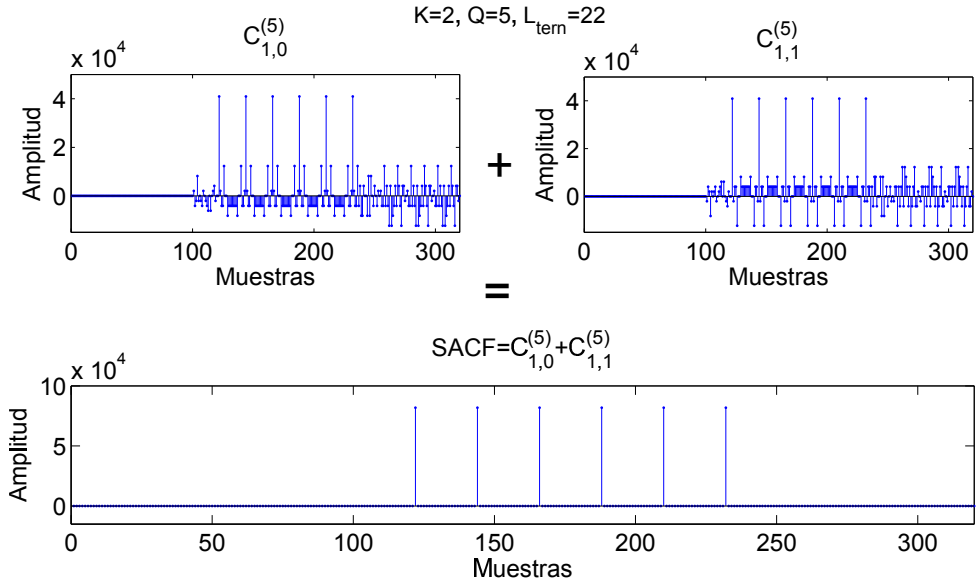
Tras implementar el diseño utilizando la opción 5 de truncamiento, en la que se consideran los bits menos significativos del dato de entrada sin contar con el bit de signo, se simularon varios esquemas en MODELSIM para validar el correcto funcionamiento del diseño. Al momento de validar un diseño, hay dos tipos de simulaciones a realizar: la simulación funcional (*behavioural* en inglés), en la que no se tiene en cuenta ningún retardo de propagación o de enrutamiento, y la simulación temporal (*Post-map and Post-route simulation* en inglés o *post PAR*) en la que se consideran los retardos de propagación, enrutamiento, así como también el retardo interno entre compuertas lógicas, además limita los recursos a los existentes en la FPGA seleccionada. Este tipo de simulación puede predecir el comportamiento del diseño en el hardware.

Para $K_{|MultCSS} = 2$, se generó un par ternario de secuencias de longitud $L_{|MultCSS} = 22$, por lo que se implementó un correlador de 5 etapas, según los parámetros de la Tabla 3.3 vista anteriormente en la sección 3.3. En la Figura 4.20, se observa una comparación de los resultados obtenidos en el modelo de simulación realizado en MATLAB de la Figura 4.20(a), con lo obtenido en la simulación funcional de la implementación hardware simulado en MODELSIM (Figura 4.20(b)). Aquí se puede ver que las simulaciones coinciden perfectamente para este caso.

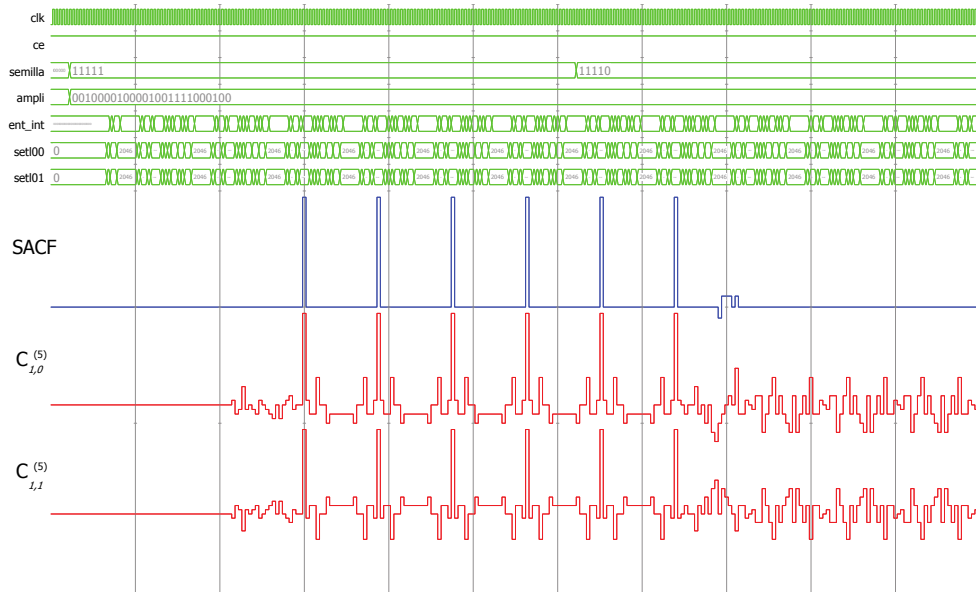
Cuando se realizan simulaciones temporales, en éstas aparecen retardos reales debido a las compuertas lógicas, como estos retardos internos se propagan a lo largo de cada bloque, produciendo variaciones de nano segundos, que conllevan la aparición de transitorios indeseados del estado final en las operaciones. Como tienen muy corta duración se pueden controlar colocando biestables (*Flip Flops*) a la salida para evitar que afecten al flanco de subida de reloj, que es cuando realmente se leen los datos para realizar la multiplicación o al final para detectar el pico de SACF. A estos transitorios se los conoce como *glitches*.

En la Figura 4.21 se puede apreciar el pico de SACF de la simulación temporal para $K_{|MultCSS} = 4$ secuencias; en ella aparecen los efectos de los transitorios, estos siempre ocurren entre flancos de subida y bajada del reloj, pero no exactamente en el flanco de subida que es cuando se lee el dato para detectar el pico de SACF, por lo que no afectan al sistema. Como se vio en la Figura 4.21, la aparición de los *glitches* no permite una correcta visualización de las gráficas de la simulación, por lo que se han almacenado en un fichero los valores correspondientes a los flancos de subida del reloj para su posterior análisis con MATLAB. En la Figura 4.22(b) se puede observar el resultado de la simulación temporal obtenida con MODELSIM, representada con MATLAB. Y en la Figura 4.22(a), lo obtenido por el modelo de simulación desarrollado. En esta comparación, se puede notar también el correcto funcionamiento de la implementación. Este ejemplo muestra además, el caso en el que la representación del dato sin truncar, supera los 25 bits, y siendo una etapa $Q = 5$ impar, el pico de correlación aparece invertido. Como se menciono anteriormente, esto se debe a que al truncar el dato, ya no se considera el bit de signo.

Por último se observa en la Figura 4.23 una comparación para $K_{|MultCSS} = 8$ secuencias. Al igual que antes, la simulación temporal es representada en MATLAB debido a la aparición de los *glitches* en este tipo de simulaciones. En la Figura 4.23(a), se puede apreciar el resultado del modelo de simulación en el que se ha concatenado la secuencia utilizando 5 periodos, en los últimos 3 se cambió la semilla de generación por la correspondiente a una de las secuencias ortogonales, por lo que la suma de las correlaciones es nula a partir de ese instante. Y en la Figura 4.23(b), la simulación temporal de la misma correlación, solo que aquí, a partir de la muestra 1500, se cambia el valor de la semilla por la de otro conjunto ortogonal, por lo que la correlación pasará a ser una correlación cruzada (CCF) y el resultado de la suma de CCF es nulo para todos los desplazamientos.



(a) Resultados del modelo de simulación de ACF y la suma de ACF de un par de secuencias complementarias ternarias con 9 periodos concatenados, simulado con MATLAB. Los últimos 3 periodos corresponden a la secuencia con la semilla ortogonal, por lo que la suma de las correlaciones es nula.



(b) ACF y suma de ACF del mismo par ternario, implementado en el correlador diseñado (simulación funcional). Una vez que se repite 6 veces la secuencia, se cambia la semilla para que el correlador tenga los parámetros del conjunto ortogonal, por lo que la suma de correlación es nula.

Figura 4.20: Par ternario obtenido a partir de los parámetros, $K_{|MultCSS} = 2$, $Q = 5$, $L_{|MultCSS} = 22$, $A^{(q)} = [1, -1/2, 1, 1, 1]$, $D^{(q)} = [4, 1, 1, 3, 12]$ y $W^{(1,q)} = [1, 1, 1, 1, 1]$, luego se cambia la semilla a $W^{(1,q)} = [-1, 1, 1, 1, 1]$.

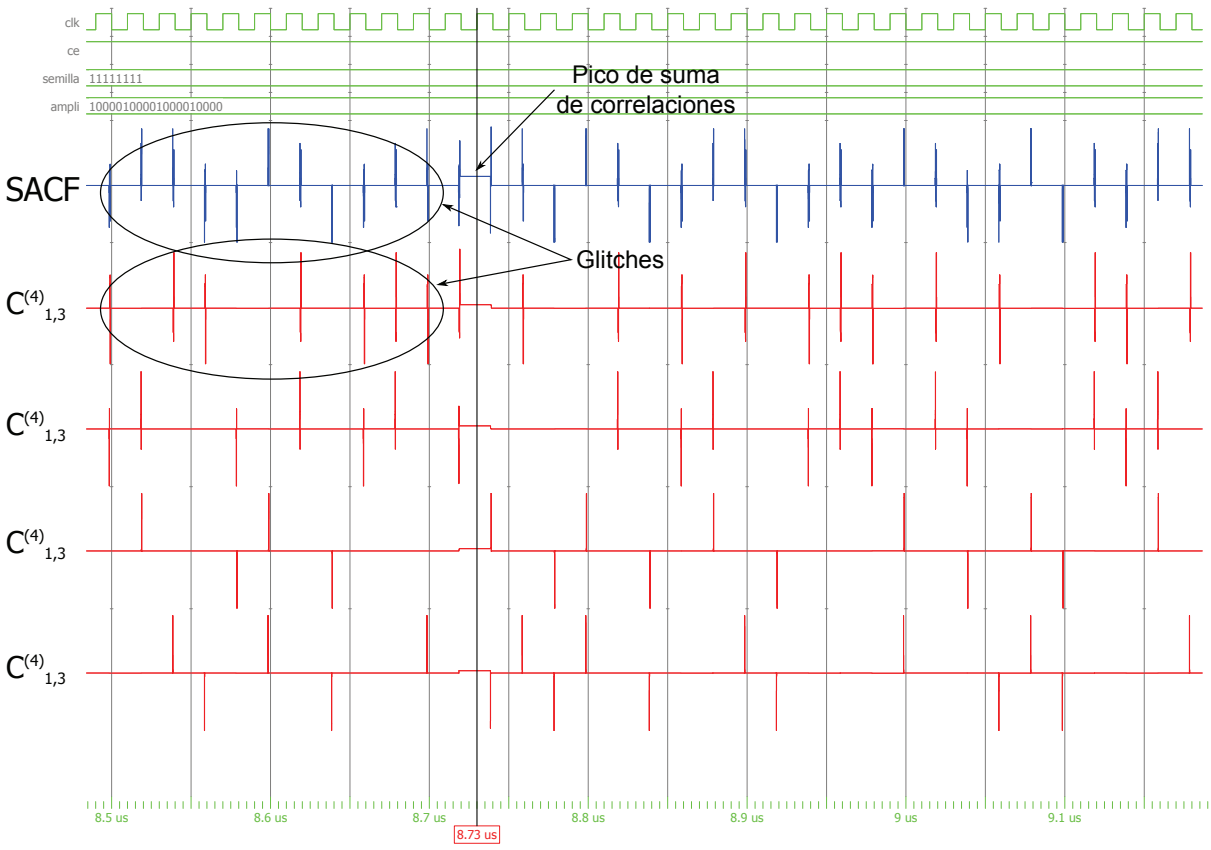
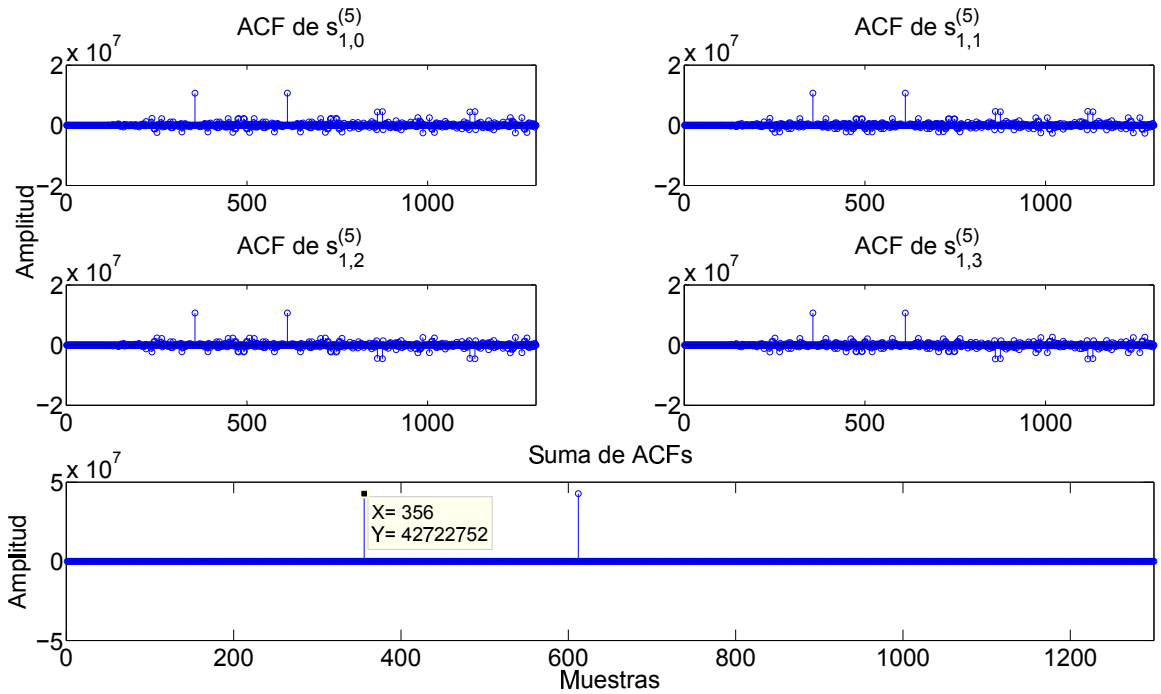
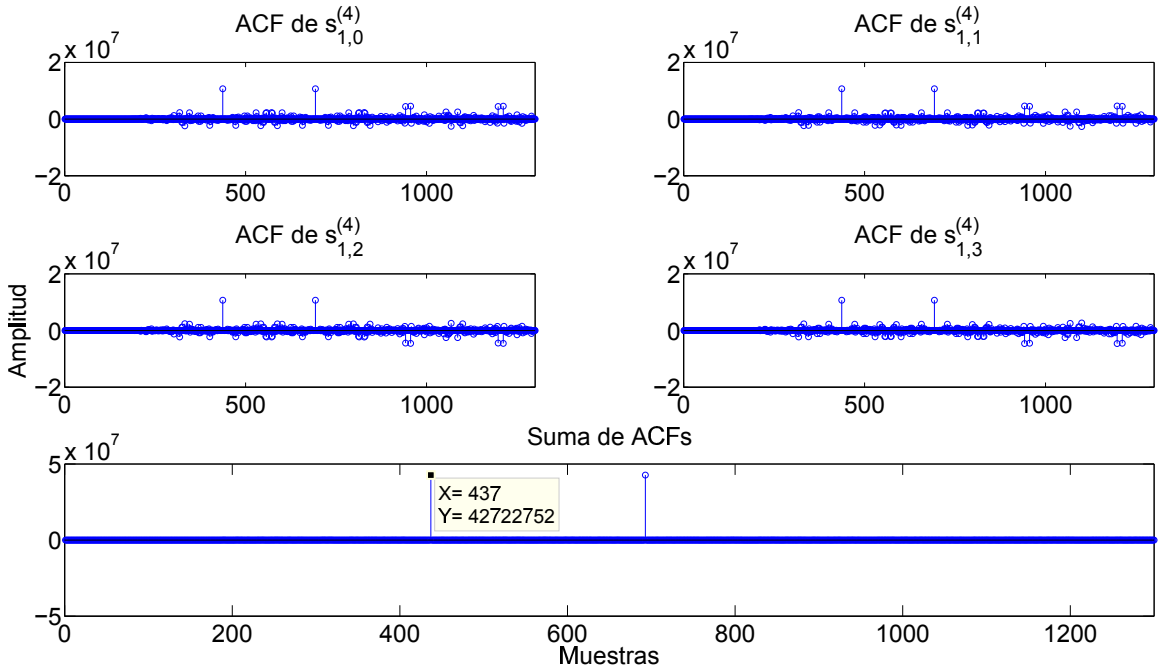


Figura 4.21: Pico de la suma de las ACFs en la simulación temporal de la correlación para $K_{|MultCSS} = 4$ secuencias. Se puede apreciar que el pico de la suma coincide con el flanco de subida del reloj clk.

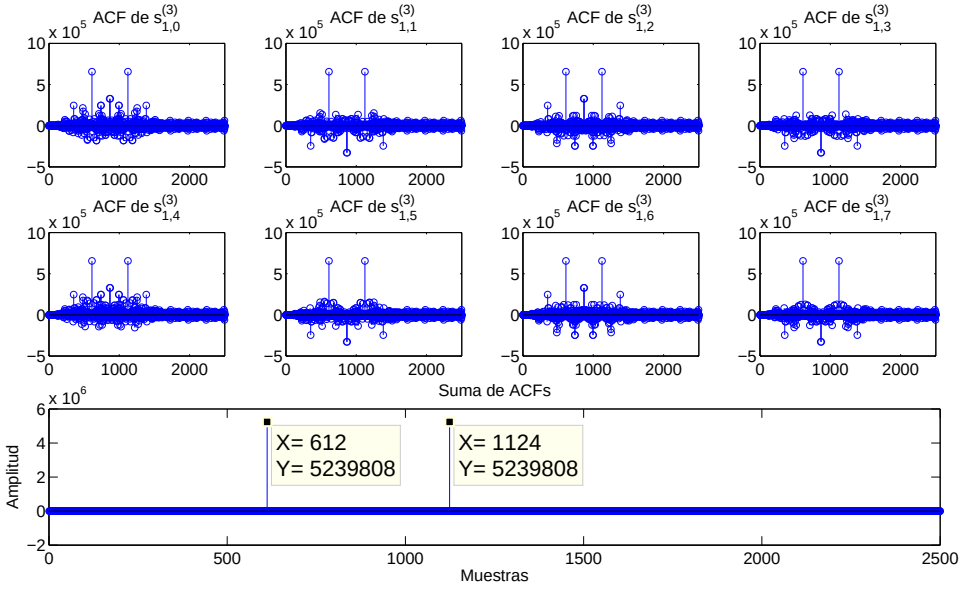


(a) Para $K_{|MultCSS} = 4$ secuencias, se ven aquí las ACFs correspondientes y la suma de ellas para 2 periodos concatenados. Correlación obtenida a partir del modelo de simulación realizado en MATLAB.

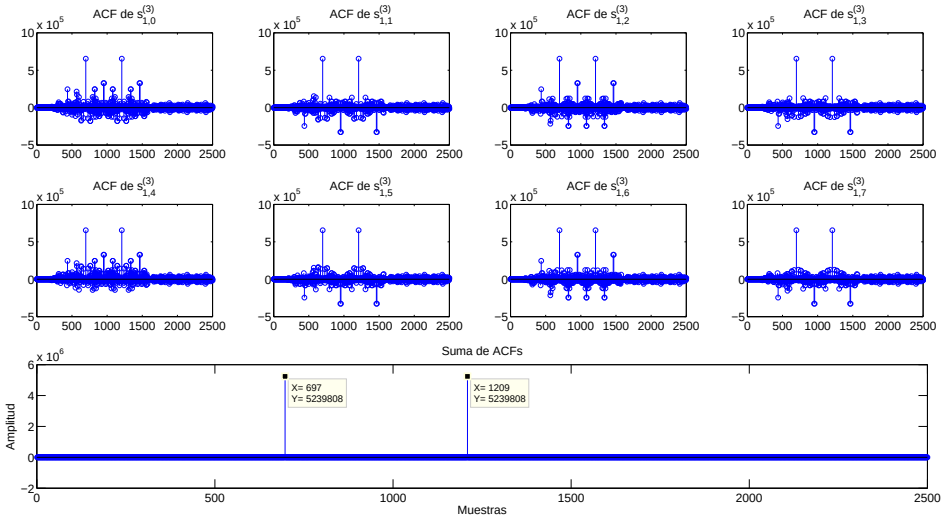


(b) Simulación temporal obtenida con MODELSIM de la implementación VHDL, pero representada con MATLAB a partir de los datos almacenados. Se representan 4 ACFs y la suma de ellas para un conjunto de $K_{|MultCSS} = 4$ secuencias.

Figura 4.22: Correlación y suma de 4 secuencias complementarias obtenido a partir de los parámetros, $K_{|MultCSS} = 4$, $Q = 4$, $L_{|MultCSS} = 256$, $A^{(q)} = [-4, -4, -4, -4]$, $D^{(q)} = [64, 16, 4, 1]$ y $W^{(1,q)} = [1, 1, 1, 1]$. Los picos de correlación negativos se deben a la opción de truncamiento elegida.



(a) Resultados del modelo de simulación de MATLAB para $K_{|MultCSS} = 8$ secuencias.



(b) Simulación temporal obtenida con MODELSIM de la implementación VHDL, pero representada con MATLAB a partir de los datos almacenados. Se representan las 8 ACFs y la suma de ellas para un conjunto de $K_{|MultCSS} = 8$ secuencias correladas. A partir de la muestra 1500 se cambia el valor de la semilla para correlar con otro conjunto ortogonal, por lo que el resultado de la suma es nulo para todos los desplazamientos.

Figura 4.23: Correlación de 8 secuencias obtenidas a partir de los parámetros, $K_{|MultCSS} = 8$, $Q = 3$, $L_{|MultCSS} = 512$, $A^{(q)} = [1, -1/2, 1]$, $D^{(q)} = [64, 8, 1]$ y $W^{(1,q)} = [1, 1, 1, 1, 1, 1, 1, 1]$, luego se cambia la semilla a $W^{(1,q)} = [-1, -1, -1, 1, 1, 1, 1, 1]$.

4.2.2. Resultados de la implementación

En las Tablas 4.5 y 4.6 se muestra el nivel de ocupación de la FPGA y la frecuencia máxima de operación del correlador, para distintos números de secuencias y longitudes de código. Para todos los casos se consideró que los retardos de cada etapa eran $D^{(q)} = \left[K^{Q-1}, K^{Q-2}, \ldots, K, K^0\right]$, con lo que las primeras etapas son las que mayor cantidad de registros de desplazamiento ocupan. Se puede notar que a medida que aumenta el número de etapas, mayor es la utilización de recursos; lo mismo sucede cuando para iguales longitudes se aumenta el número de secuencias K. Para algunos casos los recursos necesarios son superiores a los disponibles por la plataforma utilizada, por lo que estos en particular no podrían ser implementados en el modelo de FPGA elegido. Sin embargo, para las aplicaciones basadas en técnicas CDMA que se realizan en el grupo GEINTRA, estos correladores son suficientes, ya que suelen estar limitados a un máximo de 5 emisores simultáneos y longitudes en torno a 1024 bits, por lo que con un correlador de 8 secuencias bastaría para cumplir esa necesidad.

XC5VLX50T	K=2 Q=10 L=1024	K=2 Q=11 L=2048	K=2 Q=12 L=4096	K=4 Q=4 L=256	K=4 Q=5 L=1024	K=4 Q=6 L=4096
Slices	2291 7 %	2518 8 %	2745 9 %	3084 10 %	3838 13 %	4592 15 %
LUT	4214 14 %	6066 21 %	9518 33 %	4192 14 %	6571 22 %	13450 46 %
Registros de desplazamiento	2670 34 %	4370 56 %	7670 99 %	2128 27 %	4003 52 %	10378 135 %
IOBs	132 27 %	138 28 %	144 30 %	158 32 %	165 34 %	172 35 %
DSP48E	20 41 %	22 45 %	24 50 %	16 33 %	20 41 %	24 50 %
Frecuencia Max. (MHz)	221.53	222.62	108.28	179.47	174.06	—

Tabla 4.5: Ocupación de recursos en la FPGA XC5VLX50T para diferentes longitudes, de 2 y 4 secuencias.

En aquellos casos en los que no se muestra el valor de frecuencia máxima de operación en la tabla, se debe a que los recursos disponibles fueron superados para algún componente, por lo que no se puede realizar el *routeo* o implementación del diseño.

XC5VLX50T	K=8 Q=2 L=64	K=8 Q=3 L=512	K=8 Q=4 L=4096	K=16 Q=1 L=16
Slices	4028 13 %	5986 20 %	7944 27 %	4966 17 %
LUT	5256 18 %	8564 29 %	18647 64 %	6634 23 %
Registros de desplazamiento	2344 30 %	4244 55 %	12919 168 %	2826 36 %
IOBs	262 54 %	270 56 %	278 57 %	487 101 %
DSP48E	16 33 %	24 50 %	32 66 %	16 33 %
Frecuencia Max. (MHz)	185.8	157.43	—	—

Tabla 4.6: Ocupación de recursos en la FPGA XC5VLX50T para diferentes longitudes, de 8 y 16 secuencias.

Capítulo 5

Conclusiones y trabajos futuros

5.1. Conclusiones

A lo largo del presente trabajo se han presentado una variedad de arquitecturas para la generación y correlación de Conjuntos de Secuencias Complementarias binarias, y más específicamente de CSS multinivel, estos últimos tienen la particularidad de generar pares complementarios ternarios óptimos. También se han estudiado diferentes opciones de truncamiento y cuantificación, ya que el correlador CSS multinivel va creciendo en número de bits por cada etapa, por lo que se requiere limitar ese crecimiento. Específicamente, en una de las sub-etapas de la arquitectura propuesta se realiza una multiplicación que tiene un ancho fijo, por lo que se deberá en algún momento truncar el dato y por ello se ha implementado la opción de truncamiento que menos deteriore la información.

Para el estudio de las opciones de truncamiento propuestas, se analizaron en un modelo de simulación realizado en MATLAB, casos con valores de multiplicadores extremos, es decir para la representación propuesta $(-4)_{10} \equiv (100{,}00)_{CA2}$ y $(1/4)_{10} \equiv (000{,}01)_{CA2}$, y un caso más práctico donde solo uno de los multiplicadores es distinto de uno. Estos valores son extremos en el sentido que en casos prácticos, no se implementarían pero sirven para el análisis de los efectos de truncamiento. En la implementación real, para la generación de pares complementarios ternarios óptimos, solo se emplean un par de multiplicadores distintos de uno, esto para generar las secuencias ternarias, o a lo sumo de pocos niveles. De este estudio se obtuvieron varios resultados que se tabularon y graficaron para los diferentes valores de multiplicadores; comparando los efectos producidos para distintas longitudes y número de secuencias. Se utilizó como parámetro para medir la calidad de las correlaciones obtenidas, el valor de la cota o *Bound*, que brinda información de la relación de amplitud entre el pico de correlación y los lóbulos laterales.

De los resultados obtenidos, se concluyó que la mejor forma de truncar los datos a la entrada del multiplicador, en función de las opciones propuestas en este trabajo, es utilizando los bits menos significativos y truncando los de mayor peso sin tener en cuenta el bit de signo. Aunque como se vio, pueden ocurrir casos en que el pico de suma de correlaciones sea negativo, sin embargo, esto no supone ningún problema, ya que lo importante es detectar la muestra o instante de tiempo en la que se produce ese pico, y eso es fácilmente implementable con un detector de picos que realice el valor absoluto de la correlación.

Por último se verificó y validó el diseño realizando simulaciones temporales, es decir simulaciones que incluyen retardos de propagación, retardos internos entre compuertas lógicas y la limitación de los recursos según la FPGA elegida, a partir de ellas se puede estimar el funcionamiento del diseño implementado en hardware en condiciones reales. De las simulaciones temporales, se obtuvieron los mismos resultados que con el modelo de simulación realizado, con lo que se da como válido el diseño propuesto.

5.2. Trabajos futuros

Como trabajos futuros se proponen los siguientes:

- Implementar el correlador también propuesto en [GUG13] para $K_{|MultCSS} \geq 3 - \{4\}$, y combinarlo con el correlador que se planteó en este trabajo, para así poder correlar conjuntos complementarios de cualquier número de secuencias.
- Implementar los multiplicadores $A^{(q)}$ utilizando descomposición SPT (*Signed Power-of-Two*) en la que los valores de estos coeficientes se aproximan al valor teórico utilizando sumas, restas y desplazamientos.
- Implementar los retardos utilizando bloques de memoria RAM en lugar de registros de desplazamiento. Esto permitiría un ahorro en los *SLICES* utilizados, y la posibilidad de configurar de forma externa los retardos deseados, después de haber sintetizado el diseño.
- Optimizar para que el crecimiento del número de bits, sea progresivo desde el principio, y en el caso que se implementen los multiplicadores con SPT, se podrían obtener secuencias de mayor longitud, con menos errores según que coeficientes se utilicen.
- Adaptar e implementar la versión optimizada del correlador CSS binario, propuesto en [MUH+11], al alfabeto multinivel. Éste permite realizar la suma de correlaciones empleando un único correlador, en vez de K correladores requeridos con la versión implementada en este trabajo.

Anexo

6.1. Tablas con resultados de las diferentes opciones de truncamiento.

Las tablas que se presentan a continuación, contienen los resultados obtenidos de las pruebas realizadas con el modelo de simulación, en ellas se muestran los parámetros que se consideran necesarios para la validación o descarte de las 5 opciones propuestas de truncamiento. En cada columna se representa lo siguiente:

$K_{|MultCSS}$ Número de secuencias que contiene el conjunto.

Q Número de etapas del correlador, es también el parámetro con el que se generó las secuencias complementarias.

$L_{|MultCSS}$ Longitud de las secuencias correladas.

Bound SACF Es el valor de la cota o *Bound* de la suma de las ACF, este parámetro es el que define la calidad de la correlación cuanto mas pequeño sea.

pico SACF Es la amplitud del pico de la SACF obtenido, con la opción de truncamiento evaluada.

pico esperado SACF Es la amplitud que debería tener el pico de la SACF sin ningún tipo de truncamiento. Se obtiene según la ecuación (4.4).

max bit rep ACF Es el número máximo de bits que se necesitan para representar el pico de la autocorrelación obtenido, con la opción de truncamiento evaluada. Este valor no implica que a la salida del correlador, ni a la entrada de los multiplicadores se haya truncado con esta cantidad de bits.

bit rep ACF ideal Al igual que antes, representa el número máximo de bits que se necesitan para representar el pico de la ACF, pero sin aplicar ninguna opción de truncamiento. Es una referencia para saber cuando empieza a ser realmente truncado el dato, comparándolo con el dato anterior.

$A^{(q)}$ Es el valor de los Q multiplicadores utlizados para evaluar cada una de las opciones propuestas.

Tabla 6.1: Resultados para la primer opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
$A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$							
2	4	16	365.5E-6	166,891	1,335,086	17	20
2	5	32	406.1E-6	44,328	5,673,178	15	22
2	6	64	764.3E-6	11,775	24,106,320	13	24
2	7	128	2.2E-3	3,127	102,428,448	11	26
2	8	256	8.4E-3	833	435,204,096	9	28
2	9	512	27.1E-3	221	1,849,035,776	7	30
2	10	1024	67.8E-3	59	7,855,513,600	5	32
2	11	2048	235.3E-3	17	33,371,635,712	3	34
2	12	4096	500.0E-3	8	141,758,955,520	2	37
4	1	4	172.4E-6	17,815,552	34,796	23	15
4	2	16	202.9E-6	4,731,776	295,736	21	18
4	3	64	238.7E-6	1,256,728	2,513,456	19	21
4	4	256	293.6E-6	333,782	21,361,376	17	24
4	5	1024	428.6E-6	88,658	181,541,696	15	27
4	6	4096	764.1E-6	23,558	1,542,804,480	13	30
8	1	8	172.4E-6	35,631,104	69,592	23	16
8	2	64	202.9E-6	9,463,552	1,182,944	21	20
8	3	512	238.7E-6	2,513,456	20,107,648	19	24
8	4	4096	293.6E-6	667,564	341,782,016	17	28
16	1	16	172.4E-6	71,262,208	139,184	23	17
16	2	256	202.9E-6	18,927,104	4,731,776	21	22
16	3	4096	238.7E-6	5,026,912	160,861,184	19	27
$A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$							
2	4	16	1.6E-3	1,248	9,990	10	13
2	5	32	15.6E-3	96	12,485	6	13
2	6	64	214.3E-3	7	15,602	2	13
2	7	128	2.0E+0	1	19,497	1	14
2	8	256	2.0E+0	1	24,363	1	14
2	9	512	2.0E+0	1	30,441	1	14
2	10	1024	2.0E+0	1	38,032	1	15
2	11	2048	2.0E+0	1	47,511	1	15
2	12	4096	2.0E+0	1	59,345	1	15
4	1	4	97.7E-6	5,239,808	10,234	21	13
4	2	16	117.3E-6	409,312	25,582	17	14
4	3	64	125.1E-6	31,972	63,946	13	15
4	4	256	1.6E-3	2,496	159,838	10	17
4	5	1024	20.5E-3	195	399,514	6	18
4	6	4096	266.7E-3	15	998,542	2	19
8	1	8	97.7E-6	10,479,616	20,468	21	14
8	2	64	117.3E-6	818,624	102,328	17	16

Continúa...

Tabla 6.1: Resultados para la primer opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
8	3	512	125.1E-6	63,944	511,568	13	18
8	4	4096	1.6E-3	4,992	2,557,408	10	21
16	1	16	97.7E-6	20,959,232	40,936	21	15
16	2	256	117.3E-6	1,637,248	409,312	17	18
16	3	4096	125.1E-6	127,888	4,092,544	13	21
$A^{(q)} = [1\ -1/2\ 1 \ldots 1]$							
2	4	16	195.4E-6	5,117	40,936	12	15
2	5	32	3.1E-3	637	81,872	9	16
2	6	64	26.0E-3	77	163,744	6	17
2	7	128	333.3E-3	9	327,488	2	18
2	8	256	3.0E+0	1	654,976	1	19
2	9	512	3.0E+0	1	1,309,952	1	20
2	10	1024	3.0E+0	1	2,619,904	1	21
2	11	2048	3.0E+0	1	5,239,808	1	22
2	12	4096	3.0E+0	1	10,479,616	1	23
4	1	4	000.0E+0	8,384,512	16,376	21	13
4	2	16	48.9E-6	654,976	40,936	18	15
4	3	64	73.3E-6	81,872	163,744	15	17
4	4	256	586.3E-6	10,234	654,976	12	19
4	5	1024	4.7E-3	1,274	2,619,904	9	21
4	6	4096	39.0E-3	154	10,479,616	6	23
4	7	16384	333.3E-3	18	41,918,464	2	25
8	1	8	000.0E+0	16,769,024	32,752	21	14
8	2	64	48.9E-6	1,309,952	163,744	18	17
8	3	512	73.3E-6	163,744	1,309,952	15	20
8	4	4096	586.3E-6	20,468	10,479,616	12	23
16	1	16	000.0E+0	33,538,048	65,504	21	15
16	2	256	48.9E-6	2,619,904	654,976	18	19
16	3	4096	73.3E-6	327,488	10,479,616	15	23

Tabla 6.2: Resultados para la segunda opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
			$A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	1.48E+00	98942976	1335086	27	20
2	5	32	1.03E+00	136003584	5673178	27	22
2	6	64	1.25E+01	11141120	24106320	27	24
2	7	128	1.87E+00	65273856	102428448	27	26
2	8	256	1.11E+00	134217728	435204096	27	28
2	9	512	9.16E-01	159383552	1849035776	27	30
2	10	1024	1.26E+00	117440512	7855513600	27	32
2	11	2048	1.16E+00	134217728	33371635712	27	34
2	12	4096	2.28E+00	67108864	1.41759E+11	27	37
4	1	4	1.11E-01	301662208	34796	27	15
4	2	16	9.49E-01	281935872	295736	27	18
4	3	64	2.35E+00	109838336	2513456	27	21
4	4	256	3.57E+00	79167488	21361376	27	24
4	5	1024	2.87E+00	114294784	181541696	27	27
4	6	4096	2.14E+00	150994944	1542804480	27	30
8	1	8	3.00E-01	669777920	69592	27	16
8	2	64	8.20E-01	644874240	1182944	27	20
8	3	512	3.12E+00	146800640	20107648	27	24
8	4	4096	1.95E+00	251658240	341782016	27	28
16	1	16	3.00E-01	1336934400	139184	27	17
16	2	256	1.13E+00	1132462080	4731776	27	22
16	3	4096	8.00E-01	1006632960	160861184	27	27
			$A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	8.70E-02	48282624	9.99E+03	25	13
2	5	32	2.29E-01	18389504	1.25E+04	24	13
2	6	64	3.09E-01	27150080	1.56E+04	24	13
2	7	128	1.98E-01	42279552	1.95E+04	25	14
2	8	256	5.57E-01	15030720	2.44E+04	24	14
2	9	512	4.35E-01	22877728	3.04E+04	24	14
2	10	1024	2.42E-01	34731152	3.80E+04	25	15
2	11	2048	3.28E+00	3331856	4.75E+04	24	15
2	12	4096	1.91E+00	8004832	5.93E+04	24	15
4	1	4	1.67E-01	100564992	10234	25	13
4	2	16	4.01E-01	83591168	25582	25	14
4	3	64	1.63E+00	41058304	63946	24	15
4	4	256	1.90E+00	35094528	159838	25	17
4	5	1024	2.77E+00	36077568	399514	25	18
4	6	4096	5.15E+00	19103744	998542	25	19
8	1	8	1.67E-01	200933376	20468	25	14
8	2	64	1.01E+00	198967296	102328	25	16

Continúa...

Tabla 6.2: Resultados para la segunda opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
8	3	512	1.06E+00	187170816	511568	25	18
8	4	4096	1.67E+00	116391936	2557408	25	21
16	1	16	1.67E-01	401080320	40936	25	15
16	2	256	1.04E+00	383778816	409312	25	18
16	3	4096	2.27E+00	176160768	4092544	25	21
$A^{(q)} = [1\ -1/2\ 1 \ldots 1]$							
2	4	16	2.45E-04	66912256	40936	25	15
2	5	32	2.51E-01	66715648	81872	25	16
2	6	64	2.53E-01	66322432	163744	25	17
2	7	128	5.12E-01	65536000	327488	25	18
2	8	256	5.25E-01	63963136	654976	25	19
2	9	512	5.52E-01	60817408	1309952	25	20
2	10	1024	6.15E-01	54525952	2619904	25	21
2	11	2048	8.00E-01	41943040	5239808	25	22
2	12	4096	3	16777216	10479616	25	23
4	1	4	0	134086656	16376	25	13
4	2	16	1.67E-01	100270080	40936	25	15
4	3	64	5.05E-01	132644864	163744	25	17
4	4	256	1.05E+00	127926272	654976	25	19
4	5	1024	1.21E+00	109051904	2619904	25	21
4	6	4096	3.75E+00	33554432	10479616	25	23
4	7	16384	3	33554432	41918464	25	25
8	1	8	0	267911168	32752	25	14
8	2	64	3.39E-01	198180864	163744	25	17
8	3	512	1.09E+00	243269632	1309952	25	20
8	4	4096	3.75E+00	67108864	10479616	25	23
16	1	16	0	534773760	65504	25	15
16	2	256	3.56E-01	377487360	654976	25	19
16	3	4096	3.75E+00	134217728	10479616	25	23

Tabla 6.3: Resultados para la tercera opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
			$A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	1.60E-01	25	1335086	4	20
2	5	32	5.00E-01	8	5673178	2	22
2	6	64	1	4	24106320	0	24
2	7	128	Inf	0	102428448	0	26
2	8	256	Inf	0	435204096	0	28
2	9	512	Inf	0	1849035776	0	30
2	10	1024	Inf	0	7855513600	0	32
2	11	2048	Inf	0	33371635712	0	34
2	12	4096	Inf	0	1.41759E+11	0	37
4	1	4	3.68E-03	2174	34796	10	15
4	2	16	1.38E-02	578	295736	8	18
4	3	64	7.79E-02	154	2513456	6	21
4	4	256	2.40E-01	50	21361376	4	24
4	5	1024	1	18	181541696	2	27
4	6	4096	1	18	1542804480	2	30
8	1	8	3.68E-03	4348	69592	10	16
8	2	64	1.56E-02	1156	1182944	8	20
8	3	512	1.17E-01	308	20107648	6	24
8	4	4096	3.50E-01	100	341782016	4	28
16	1	16	3.68E-03	8696	139184	10	17
16	2	256	2.34E-02	2312	4731776	8	22
16	3	4096	1.17E-01	616	160861184	6	27
			$A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	2	1	9.99E+03	1	13
2	5	32	2	1	1.25E+04	1	13
2	6	64	2	1	1.56E+04	1	13
2	7	128	2	1	1.95E+04	1	14
2	8	256	2	1	2.44E+04	1	14
2	9	512	2	1	3.04E+04	1	14
2	10	1024	2	1	3.80E+04	1	15
2	11	2048	2	1	4.75E+04	1	15
2	12	4096	2	1	5.93E+04	1	15
4	1	4	3.14E-03	636	10234	8	13
4	2	16	4.35E-02	46	25582	4	14
4	3	64	1.50E+00	2	63946	1	15
4	4	256	1.75E+00	2	159838	1	17
4	5	1024	2	2	399514	1	18
4	6	4096	2	2	998542	1	19
8	1	8	3.14E-03	1272	20468	8	14
8	2	64	4.35E-02	92	102328	4	16

Continúa...

Tabla 6.3: Resultados para la tercera opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
8	3	512	1.75E+00	4	511568	1	18
8	4	4096	1.75E+00	4	2557408	1	21
16	1	16	3.14E-03	2544	40936	8	15
16	2	256	4.35E-02	184	409312	4	18
16	3	4096	1.88E+00	8	4092544	1	21
$A^{(q)} = [1 \ -1/2 \ 1 \ldots 1]$							
2	4	16	2	1	40936	1	15
2	5	32	3	1	81872	1	16
2	6	64	3	1	163744	1	17
2	7	128	3	1	327488	1	18
2	8	256	3	1	654976	1	19
2	9	512	3	1	1309952	1	20
2	10	1024	3	1	2619904	1	21
2	11	2048	3	1	5239808	1	22
2	12	4096	3	1	10479616	1	23
4	1	4	2.95E-03	1018	16376	8	13
4	2	16	3.85E-02	78	40936	5	15
4	3	64	5.00E-01	10	163744	1	17
4	4	256	3	2	654976	1	19
4	5	1024	3	2	2619904	1	21
4	6	4096	3	2	10479616	1	23
4	7	16384	3	2	41918464	1	25
8	1	8	2.95E-03	2036	32752	8	14
8	2	64	2.56E-02	156	163744	5	17
8	3	512	5.50E-01	20	1309952	1	20
8	4	4096	3	4	10479616	1	23
16	1	16	2.95E-03	4072	65504	8	15
16	2	256	2.56E-02	312	654976	5	19
16	3	4096	5.75E-01	40	10479616	1	23

Tabla 6.4: Resultados para la cuarta opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
			$A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	3.93E-04	1335086	1335086	20	20
2	5	32	4.63E-04	5673178	5673178	22	22
2	6	64	5.44E-04	24106320	24106320	24	24
2	7	128	6.39E-04	102428448	102428448	26	26
2	8	256	5.17E-01	32550912	435204096	26	28
2	9	512	2.45E-01	137759744	1849035776	27	30
2	10	1024	2.69E+00	37330944	7855513600	27	32
2	11	2048	4.96E+00	18530304	33371635712	27	34
2	12	4096	1.62E+00	92143616	1.41759E+11	27	37
4	1	4	1.72E-04	69592	34796	15	15
4	2	16	2.03E-04	591472	295736	18	18
4	3	64	2.39E-04	5026912	2513456	21	21
4	4	256	2.81E-04	42722752	21361376	24	24
4	5	1024	1.27E-03	94647936	181541696	25	27
4	6	4096	2.51E-01	267036672	1542804480	26	30
8	1	8	1.72E-04	278368	69592	16	16
8	2	64	2.03E-04	4731776	1182944	20	20
8	3	512	2.39E-04	80430592	20107648	24	24
8	4	4096	2.28E-01	293386240	341782016	26	28
16	1	16	1.72E-04	1113472	139184	17	17
16	2	256	2.03E-04	37854208	4731776	22	22
16	3	4096	1.44E-03	213147648	160861184	25	27
			$A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	2.50E-04	9990	9.99E+03	13	13
2	5	32	3.60E-04	12484	1.25E+04	13	13
2	6	64	3.52E-04	15604	1.56E+04	13	13
2	7	128	4.36E-04	1.95E+04	1.95E+04	14	14
2	8	256	4.31E-04	2.44E+04	2.44E+04	14	14
2	9	512	4.60E-04	30444	3.04E+04	14	14
2	10	1024	5.26E-04	3.80E+04	3.80E+04	15	15
2	11	2048	5.37E-04	47520	4.75E+04	15	15
2	12	4096	6.23E-04	59364	5.93E+04	15	15
4	1	4	9.77E-05	20468	10234	13	13
4	2	16	1.17E-04	51164	25582	14	14
4	3	64	1.41E-04	127892	63946	15	15
4	4	256	1.69E-04	319676	159838	17	17
4	5	1024	2.03E-04	799028	399514	18	18
4	6	4096	2.43E-04	1997084	998542	19	19
8	1	8	9.77E-05	81872	20468	14	14
8	2	64	1.17E-04	409312	102328	16	16

Continúa...

Tabla 6.4: Resultados para la cuarta opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
8	3	512	1.41E-04	2046272	511568	18	18
8	4	4096	1.69E-04	10229632	2557408	21	21
16	1	16	9.77E-05	327488	40936	15	15
16	2	256	1.17E-04	3274496	409312	18	18
16	3	4096	1.41E-04	32740352	4092544	21	21
$A^{(q)} = [1\ -1/2\ 1 \ldots 1]$							
2	4	16	4.89E-05	40936	40936	15	15
2	5	32	4.89E-05	81872	81872	16	16
2	6	64	3.66E-05	163744	163744	17	17
2	7	128	4.89E-05	327488	327488	18	18
2	8	256	4.27E-05	654976	654976	19	19
2	9	512	4.89E-05	1309952	1309952	20	20
2	10	1024	4.58E-05	2619904	2619904	21	21
2	11	2048	4.89E-05	5239808	5239808	22	22
2	12	4096	4.73E-05	10479616	10479616	23	23
4	1	4	0	32752	16376	13	13
4	2	16	4.89E-05	81872	40936	15	15
4	3	64	4.89E-05	327488	163744	17	17
4	4	256	4.89E-05	1309952	654976	19	19
4	5	1024	4.89E-05	5239808	2619904	21	21
4	6	4096	4.89E-05	20959232	10479616	23	23
4	7	16384	4.89E-05	83836928	41918464	25	25
8	1	8	0	131008	32752	14	14
8	2	64	4.89E-05	654976	163744	17	17
8	3	512	4.89E-05	5239808	1309952	20	20
8	4	4096	4.89E-05	41918464	10479616	23	23
16	1	16	0	524032	65504	15	15
16	2	256	4.89E-05	5239808	654976	19	19
16	3	4096	4.89E-05	83836928	10479616	23	23

Tabla 6.5: Resultados para la quinta opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
			$A^{(q)} = -4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	3.93E-04	1335086	1335086	20	20
2	5	32	4.63E-04	5673178	5673178	22	22
2	6	64	5.44E-04	24106320	24106320	24	24
2	7	128	6.39E-04	102428448	102428448	26	26
2	8	256	3.30E-01	-101666816	435204096	26	28
2	9	512	6.40E-01	104205312	1849035776	26	30
2	10	1024	1.04E+00	-130441216	7855513600	27	32
2	11	2048	6.93E+00	18530304	33371635712	27	34
2	12	4096	1.56E+00	92143616	1.41759E+11	27	37
4	1	4	1.72E-04	69592	34796	15	15
4	2	16	2.03E-04	591472	295736	18	18
4	3	64	2.39E-04	5026912	2513456	21	21
4	4	256	2.81E-04	42722752	21361376	24	24
4	5	1024	6.90E-04	-173787520	181541696	26	27
4	6	4096	2.52E-01	267036672	1542804480	26	30
8	1	8	1.72E-04	278368	69592	16	16
8	2	64	2.03E-04	4731776	1182944	20	20
8	3	512	2.39E-04	80430592	20107648	24	24
8	4	4096	4.57E-01	293386240	341782016	26	28
16	1	16	1.72E-04	1113472	139184	17	17
16	2	256	2.03E-04	37854208	4731776	22	22
16	3	4096	3.57E-04	-860594176	160861184	26	27
			$A^{(q)} = 1/4 \cdot [1\ 1\ \ldots 1]$				
2	4	16	2.50E-04	9990	9.99E+03	13	13
2	5	32	3.60E-04	12484	1.25E+04	13	13
2	6	64	3.52E-04	15604	1.56E+04	13	13
2	7	128	4.36E-04	1.95E+04	1.95E+04	14	14
2	8	256	4.31E-04	2.44E+04	2.44E+04	14	14
2	9	512	4.60E-04	30444	3.04E+04	14	14
2	10	1024	5.26E-04	3.80E+04	3.80E+04	15	15
2	11	2048	5.37E-04	47520	4.75E+04	15	15
2	12	4096	6.23E-04	59364	5.93E+04	15	15
4	1	4	9.77E-05	20468	10234	13	13
4	2	16	1.17E-04	51164	25582	14	14
4	3	64	1.41E-04	127892	63946	15	15
4	4	256	1.69E-04	319676	159838	17	17
4	5	1024	2.03E-04	799028	399514	18	18
4	6	4096	2.43E-04	1997084	998542	19	19
8	1	8	9.77E-05	81872	20468	14	14
8	2	64	1.17E-04	409312	102328	16	16

Continúa...

Tabla 6.5: Resultados para la quinta opción de truncamiento propuesta.

$K_{\|MultCSS}$	Q	$L_{\|MultCSS}$	Bound SACF	pico SACF	pico esperado SACF	max bit rep ACF	bit rep ACF ideal
8	3	512	1.41E-04	2046272	511568	18	18
8	4	4096	1.69E-04	10229632	2557408	21	21
16	1	16	9.77E-05	327488	40936	15	15
16	2	256	1.17E-04	3274496	409312	18	18
16	3	4096	1.41E-04	32740352	4092544	21	21
$A^{(q)} = [1\ -1/2\ 1 \ldots 1]$							
2	4	16	4.89E-05	40936	40936	15	15
2	5	32	4.89E-05	81872	81872	16	16
2	6	64	3.66E-05	163744	163744	17	17
2	7	128	4.89E-05	327488	327488	18	18
2	8	256	4.27E-05	654976	654976	19	19
2	9	512	4.89E-05	1309952	1309952	20	20
2	10	1024	4.58E-05	2619904	2619904	21	21
2	11	2048	4.89E-05	5239808	5239808	22	22
2	12	4096	4.73E-05	10479616	10479616	23	23
4	1	4	0	32752	16376	13	13
4	2	16	4.89E-05	81872	40936	15	15
4	3	64	4.89E-05	327488	163744	17	17
4	4	256	4.89E-05	1309952	654976	19	19
4	5	1024	4.89E-05	5239808	2619904	21	21
4	6	4096	4.89E-05	20959232	10479616	23	23
4	7	16384	4.89E-05	83836928	41918464	25	25
8	1	8	0	131008	32752	14	14
8	2	64	4.89E-05	654976	163744	17	17
8	3	512	4.89E-05	5239808	1309952	20	20
8	4	4096	4.89E-05	41918464	10479616	23	23
16	1	16	0	524032	65504	15	15
16	2	256	4.89E-05	5239808	654976	19	19
16	3	4096	4.89E-05	83836928	10479616	23	23

Bibliografía

[ÁHU+06] F. J. Álvarez, A. Hernández, J. Ureña, M. Mazo, J. J. García, J. A. Jiménez y A. Jiménez. Real-time implementation of an efficient correlator for complementary sets of four sequences applied to ultrasonic pulse compression systems. *Microprocessors and Microsystems*, 30 (1): 43–51, 2006.

[ÁUM+04] F. J. Álvarez, J. Ureña, M. Mazo, A. Hernández, J. J. García y J. A. Jiménez. Efficient generator and pulse compressor for complementary sets of four sequences. *Electronics Letters*, 40 (11): 703–704, 2004.

[Bud89] S. Z. Budisin. Fast PN sequence correlation by using FWT. *Proceedings of the 1989 Mediterranean Electrotechnical Conference, MELECON'89*, pages 513–515, 1989.

[Bud90] S. Z. Budisin. New multilevel complementary pairs of sequences. *Electronics Letters*, 26 (3): 1861–1863, 1990.

[Bud91] S. Z. Budisin. Efficient pulse compressor for golay complementary sequences. *Electronics Letters*, 27 (3): 219–220, 1991.

[CPG+13] J. M. Castilla, M. C. Pérez, E. García, R. García, J. Ureña y J. J. García. Implementación Hardware de un Correlador Eficiente para Parejas Golay Derivadas de Kernels de Longitud 2, 10 y 26. En *Seminario anual de automática, electrónica industrial e instrumentación, SAEEI 2013*, volume 1, pages 1–6, Madrid, España, Julio 10-13, 2013.

[DJ99] J. A. Davis y L. Jedwab. Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed-Muller codes. *IEEE Transactions on Information Theory*, 45 (7): 2397–2417, Noviembre 1999.

[DK88] M. Darnell y A. H. Kemp. Synthesis of multilevel complementary sequences. *Electronics Letters*, 24 (19): 1251–1252, Septiembre 1988.

[EDA11] EDA Fun. What's fpga? `http://www.edafun.com/home/item/77-whats-fpga?.html`, 2011.

[FD96] P. Fan y M. Darnell. *Sequence design for communications applications*. Research Studies Pre, 1996.

[Gar13] E. García. *Efficient complementary sequences-based architectures and their application to ranging measurements*. Tesis doctoral, Univ. de Alcalá. Departamento de Electrónica, Alcalá de Henares, España, 2013.

[GL94] A. Gavish y A. Lempel. On ternary complementary sequences. *IEEE Transactions on Information Theory*, 40 (2): 522–526, 1994.

[Gol61] M. J. E. Golay. Complementary series. *IRE Transactions on Information Theory*, 7 (2): 82–87, 1961.

[Gol67] R. Gold. Optimal binary sequences for spread spectrum multiplexing. *IEEE Transactions on Information Theory*, 13 (4): 619–621, Octubre 1967.

[GS01] M. Gysin y J. Seberry. On ternary complementary pairs. *Australasian Journal of Combinatorics*, 23: 153–170, 2001.

[GUG+12] E. García, J. Ureña, J. J. García, M. C. Pérez y D. Ruiz. Efficient generator/correlator of GPC sequences for QS-CDMA. *IEEE Communication Letters*, 16 (10): 1676–1679, Octubre 2012.

[GUG13] E. García, J. Ureña y J. J. García. Generation and correlation architectures of multilevel complementary sets of sequences. *IEEE Transactions on Signal Processing*, 61 (24): 6333–6343, 2013.

[HG88] G. Hayward y Y. Gorfu. A digital hardware correlation system for fast ultrasonic data acquisition in peak power limited applications. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 35 (6): 800–808, Noviembre 1988.

[HUH+03] A. Hernández, J. Ureña, D. Hernanz, M. Mazo, J. J. García, J. P. Dérutin, J. Serot y S. E. Palazuelos. Real-time implementation of an efficient golay correlator (EGC) applied to ultrasonic sensorial systems. *Microprocessors and Microsystems*, 27 (8): 397–406, 2003.

[HWY+11] L. He, Z. Wang, F. Yang, S. Chen y L. Hanzo. Preamble design using embedded signaling for OFDM broadcast systems based on reduced-complexity distance detection. *IEEE Transactions on Vehicular Technology*, 60 (3): 1217–1222, Marzo 2011.

[Kas66] T. Kasami. Weight distribution formula for some class of cyclic codes. Technical Report AD0632574, Coordinated Science Lab. University of Illinois, Abril 1966.

[KD82] A. H. Kemp y M. Darnell. Synthesis of uncorrelated and nonsquare sets of multilevel complementary sequences. *Electronics Letters*, 25 (12): 791–792, Junio 1982.

[LCZ10] S. F. Li, J. Chen y L. Q. Zhang. Optimisation of complete complementary codes in MIMO radar system. *Electronics Letters*, 46 (16): 1157–1159, Agosto 2010.

[MUH+06] C. De Marziani, J. Ureña, A. Hernández, M. Mazo, J. J. García, A. Jiménez, J. M. Villadangos, M. C. Pérez, A. Ochoa, y F. J. Álvarez. Inter-symbol interference reduction on macro-sequences generated from complementary set of sequences. *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, pages 3367–3372, 2006.

[MUH+07] C. De Marziani, J. Ureña, A. Hernández, M. Mazo, F. J. Álvarez, J. J. García y P. Donato. Modular architecture for efficient generation and correlation of complementary set of sequences. *IEEE Transactions on Signal Processing*, 55 (5): 2323–2337, 2007.

[MUH+11] C. De Marziani, J. Ureña, A. Hernández, J. J. García, F. J. Álvarez, Jiménez A y M. C. Pérez. Recursive algorithm to directly obtain the sum of correlations in a CSS. *IEEE Transactions on Signal Processing*, 91 (5): 1343–1346, 2011.

[Pér09] M. C. Pérez. *Generación y correlación eficiente de códigos binarios derivados de conjuntos de secuencias complementarias para sistemas ultrasónicos*. Tesis doctoral, Univ. de Alcalá. Departamento de Electrónica, Alcalá de Henares, España, 2009.

[Pop97] B. M. Popovic. Efficient despreaders for multi-code cdma systems. *Proceedings of IEEE Universal Personal Communication Record*, 2: 516–520, Octubre 1997.

[Pop99] B. M. Popovic. Efficient golay correlator. *Electronics Letters*, 35 (17): 1427–1428, Agosto 1999.

[PUH+06] M. C. Pérez, J. Ureña, A. Hernández, C. De Marziani, A. Ochoa y W. P. Marnane. FPGA implementation of an efficient correlator for complementary sets of sequences. En *Proc. of IEEE International Conference on Field Programmable Logic and*

Applications (FPL'06), pages 697–700, Madrid, España, Agosto 2006.

[PUH+07] M. C. Pérez, J. Ureña, A. Hernández, C. De Marziani, C. Jiménez y W. P. Marnane. Hardware implementation of an efficient correlator for interleaved complementary sets of sequences. *Journal of Universal Computer Sciences*, 13 (3): 388–406, 2007.

[PUH+09] M. C. Pérez, J. Ureña, A. Hernández, C. De Marziani y A. Jiménez. Efficient generation and correlation of sequence pairs with three zero-correlation zones. *IEEE Transactions on Signal Processing*, 57 (9): 3450–3465, Abril 2009.

[Rud59] W. Rudin. Some theorems on fourier coefficients. *Proceedings of the American Mathematical Society*, 10: 855–859, 1959.

[SBH01] S. Stanczak, H. Boche y M. Haardt. Are LAS-codes a miracle? En *Global Telecommunications Conference. GLOBECOM '01*, volume 1, pages 589–593, San Antonio, TX, USA, Noviembre 25-29, 2001.

[Sha51] H. S. Shapiro. Extremal problems for polynomials and power series. Master's thesis, Massachusetts Institute of Technology, 1951.

[Syl67] J. J. Sylvester. Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tesselated pavements in two or more colours, with applications to Newtonś rule, ornamental tile-work, and the theory of numbers. *Phylosophical Magazine*, 34: 461–475, 1867.

[TL72] C. C. Tseng y C. Liu. Complementary sets of sequences. *IEEE Transactions on Information Theory*, 18 (5): 644–652, 1972.

[WC92] J. D. H. White y R. E. Challis. A golay sequencer based ndt system for highly attenuating materials. *IEE Colloquium on Non-Contacting and Remote NDT*, pages 7/1 – 7/7, 1992.

[Wel74] L. R. Welch. Lower bounds on the maximum cross-correlation of signals. *IEEE Transactions on Information Theory*, 20 (3): 397–399, Mayo 1974.

[Wik07] Wikibooks. Programmable logic/fpgas. `http://en.wikibooks.org/wiki/Programmable_Logic/FPGAs`, 2007.

[Wor95] G. W. Wornell. Spreading signature CDMA: Efficient multiuser communication in the presence of fading. *IEEE Transactions on Information Theory*, 41 (5): 1418–1438, 1995.

[Xil08] Xilinx. Virtex-4 FPGA User Guide - UG070(v2.6). `http://www.xilinx.com/support/documentation/user_guides/ug070.pdf`, 2008.

[Xil12] Xilinx. Virtex-5 FPGA XtremeDSP Design Considerations User Guide - UG193 (v3.5), 2012.